30th Annual INCOSE international symposium
Virtual Event
July 20 - 22, 2020

Achieving System-of-Systems Interoperability Levels Using Linked Data and Ontologies

Jakob Axelsson
Mälardalen University
PO Box 883, SE-721 23 Västerås, Sweden
jakob.axelsson@mdh.se

Abstract. Interoperability is a key concern in systems-of-systems (SoS). Numerous frameworks have been proposed to deal with this, but they are generally on a high level and do not provide specific guidance for technical implementation. However, in the context of simulation, the Levels of Conceptual Interoperability Model (LCIM) has been proposed. Also, the semantic web initiative has been introduced to provide description logic information to web pages. This paper investigates how these two concepts can be combined into a general approach for SoS interoperability. It also expands on the LCIM model by providing more details about the world models of a system and its content on the higher levels of interoperability. The combination is illustrated using an example of autonomous vehicles, and experiences from other applications are also discussed.

Introduction

A well-known challenge (or "pain point") in systems-of-systems (SoS) engineering is how to effectively integrate the operationally and managerially independent constituent systems (CS). In particular, it is a challenge to create and maintain interoperability as the SoS evolves over time (Dahmann, 2014). Several studies on important aspects treated in the SoS engineering (SoSE) literature also highlight interoperability as one of the key concerns, such as Axelsson (2015) who studied the literature on SoSE in general and identified interoperability as the second most commonly mentioned property; Klein and van Vliet (2013) who surveyed 200 papers on SoS architecture and identified interoperability as the top concern; and Bianchi et al. (2015) who concluded from a survey of 40 papers that interoperability is the most commonly considered quality attribute.

Interoperability has been defined as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" (IEEE, 1990). The concept thus clearly goes beyond simple information transfer which is to a large extent covered by the Open Systems Interconnect (OSI) model (ITU-T, 1994). Interoperability implies that the information can be used in some meaningful way, such as letting the CS fulfil their own objectives or effectively contributing to the SoS goals, and this requires at least some interpretation and contextualization of the data.

Several frameworks have been developed to improve interoperability in general. These include the European Interoperability Framework (EIF) which gives general recommendations, such as striving for openness, and to follow standards. It also describes four levels of interoperability, namely technical; semantic; organizational; and legal. Another framework is the NATO Interoperability Standards and Profiles (NISP) which essentially provides an extensive list of standards to be followed. Finally, in the US, a National Information Exchange Model (NIEM) has been developed which proposes common vocabularies, both for recurring core concepts and domain specific concepts.

Although there is a value in having these kinds of frameworks as high-level recommendations on how to improve interoperability in a specific context, they do not provide coherent frameworks for dealing with SoS interoperability in general, nor do they provide concrete guidance. However, in the context of distributed, federated simulations, an approach called the Levels of Conceptual Interoperability Model (LCIM) has been proposed (Tolk et al, 2006; Wang et al, 2009). This appears to be a suitable model for analyzing interoperability also for SoS in general.

Tolk et al. (2006) recognize the importance of ontological concepts when using LCIM in practice. A widely used approach for describing ontologies is the semantic web initiative (Berners-Lee et al., 2001). It is based on description logic concepts and was originally developed to allow web pages to express machine-understandable meaning, and not just visual elements. It is a highly universal and extensible approach which allows the creation of terminology and ontologies that can be used just as well in other systems than the World Wide Web (WWW).

One might assume that interoperability is always good and that it should be maximized. However, as most other properties it comes at a cost in terms of engineering, performance, etc. Therefore, a trade-off must be made to find the appropriate level. LCIM provides guidance on what functionality can be achieved on each level of interoperability, and the semantic web technologies provide solutions at each level that can be assessed by systems engineers to understand the trade-off. We believe that this combination provides a highly useful approach to SoS interoperability, and the contribution of this paper is to detail how this can be done, and some of its implications. To our knowledge, this combination has not been presented before, and the paper thereby provides useful guidance for SoS researchers and engineers.

The remainder of the paper is structured as follows: In the next paragraph, LCIM and the semantic web techniques are introduced in more detail as they constitute the frame of reference for the research. Then, it is discussed for each of the interoperability levels in LCIM what elements of the semantic web are needed, and what kind of information needs to be exchanged and stored. The discussion is illustrated through a running example of autonomous vehicles which are the CS of a transportation SoS. This is followed by a discussion focusing on some of our experiences of using the proposed techniques in practice. The final two sections discuss some further related work and summarize the conclusions.

Frames of reference

This paper is using two important frames of reference, namely LCIM and the semantic web, and they will now be introduced in more detail.

Levels of Conceptual Interoperability Model (LCIM)

LCIM has been presented in several slightly different versions over the years, and the presentation here is based on what appears to be the most recent ones (Tolk et al., 2006; Wang et al., 2009). The model consists of seven levels, numbered from 0 to 6, which are summarized in Table 1. Assuming two systems S1 and S2, the levels can be described as follows:

0. *None*: Stand-alone systems which have no interoperability. S1 and S2 are unable to exchange any information at all.
1. *Technical*: A communication protocol exists for exchanging data between participating systems. A communication infrastructure is established allowing it to exchange bits and bytes, and the underlying networks and protocols are unambiguously defined. It is thus possible to reliably exchange a bit sequence between S1 and S2.

Table 1: Levels of conceptual interoperability.

Level	Layer	Premise	Information defined
6	Conceptual	Conceptual model	Assumptions, constraints
5	Dynamic	Execution model	Effect of data
4	Pragmatic	Workflow model	Use of data
3	Semantic	Reference model	Meaning of data
2	Syntactic	Data structure	Structure of data
1	Technical	Communication protocol	Bits and bytes
0	None	No connection	None

2. *Syntactic*: The format of information exchange is unambiguously defined. If faced with the same bit sequence, S1 and S2 will extract the same information entities (exchangeable symbols).

3. *Semantic*: A common information exchange reference model is used. The meaning of data terms is shared, and the content of information exchange requests are unambiguously defined. S1 and S2 agree on what terms are to be used for different information items and will draw the same conclusions from a given data set.

4. *Pragmatic*: The context where the information is used, i.e. the system states and processes, is understood by the participating systems, and is unambiguously defined. The meaning of the terms is put in a context and it is explained how the terms are used to represent functions and parameters. S1 has a model of what states and functions S2 contains, and vice versa, and they know how to request services from each other.

5. *Dynamic*: As a system operates on data over time, the state of that system will change, and this includes the assumptions and constraints that affect its data interchange. On this level, the systems are able to comprehend the state changes that occur. S1 has a model of the state transitions that can happen in S2 in response to data, and vice versa.

6. *Conceptual*: The conceptual models, i.e. the assumptions and constraints of the meaningful abstractions of reality, are aligned. This means that S1 and S2 are aware of each other's information, processes, contexts, and modeling assumptions.

Level 1 is dealing with *integrability*, with a focus on network connectivity; levels 2-4 with *interoperability*, with an implementation focus; and levels 5-6 with *composability*, where modeling abstractions are emphasized.

The model can be used both descriptively, to show what gaps in interoperability exists in a given SoS, and prescriptively, to provide requirements on the CS to achieve interoperability.

Tolk et al. (2006) describe how ontologies, which they define as a formal specification of a conceptualization, are needed to reach the highest level. They also specify what elements must be possible to represent in a formalism, including entities that represent "things" both at an instance and type level and that have unique names; relations between entities and between instance and type levels; properties and their values for the entities; and rules that define grammars of information exchange. As will be seen next, these needs are all fulfilled by the semantic web techniques. However, an explanation of LCIM in terms of the semantic web does not appear in previous publications.

Semantic web initiative

The semantic web concepts will here be used as a foundation for SoS information representation. The main building blocks are a generic data representation called the Resource Description Framework (RDF) that uses 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.

RDF is the WWW "framework for expressing information about resources, where a resource can be anything, including documents, people, physical objects, and abstract concepts" (W3C, 2014). A resource is thus the basic information entity and they represent "things" that the systems need to reason about, where the "things" can be both concrete and abstract concepts.

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. Although this is just a convention, it ensures if followed that there will not be conflicts where different entities are referred to by the same name.

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

Many of the ontological concepts of the semantic web are similar to the ones used in UML and inherited in SysML (although the terminology used in the latter is a bit more remote from the common origins in description logic.) However, whereas UML and SysML conceptual models are to a large extent motivated by the need to communicate information between engineers developing a system, and thus have a focus on human readability, the semantic web concepts are mainly used for machine readability. In an implementation, the RDF data is often stored in databases in the involved systems, but there are also several textual formats that have various degrees of human readability, and these formats can be used for information transfer between systems and between humans.

A key aspect of semantic web data is its openness. There is not a fixed and limited schema describing which data is allowed in a certain context, but the ontology is extensible. This is a key aspect for its utilization in SoS, since evolution is one of the defining characteristics of such systems.

Achieving levels of SoS interoperability using semantic web techniques

It will now be analyzed how the semantic web techniques can be applied at each of the interoperability levels (ignoring level 0, for obvious reasons). It is also discussed what data is necessary to represent at each level, and what implications this has.

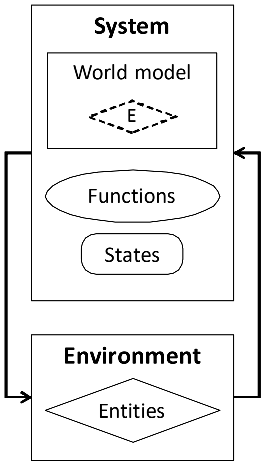

Figure 1. A basic system model.

Example

To illustrate the concepts at the various levels, an example of autonomous vehicles is used. The vehicles are the CS of a transportation SoS, which can be seen as a collaborative SoS. The vehicles try to navigate to their destinations safely without collisions, based on data from their sensors and from static sources such as maps. However, such a non-interoperable set-up where vehicles only rely on themselves is not likely to be efficient since the vehicles have to be very cautious in order to avoid accidents. By instead taking an SoS approach, where vehicles exchange information with each other and with mediating systems such as traffic control centers, traffic lights, etc., the vehicles will have access to information which they cannot obtain in other ways, allowing them to make better decision. This requires interoperability, and the example will illustrate in terms of RDF some of the advantages that can be achieved as the ladder of LCIM is climbed.

A basic system model

Before going into the LCIM levels in more detail, we will introduce a simple abstract model to explain what aspects of a system to consider. We have found this model useful to further understand LCIM. The model is shown in Figure 1, and it consists of a system and an environment. The environment contains a number of entities that can to some degree be observed and affected by the system through interactions. The system internally consists of three parts: states, functions, and a world model. The states represent internal information about the system itself, such as modes. The world model contains information about the outside, and in this simple case it is the entities of the environment that need to be represented. The world model thus stores observed or derived data about the environment that it finds necessary to keep track of. The functions determine what outputs the system will generate, what state changes occur, and what updates are needed to the world model. This is all decided based on the current state, the current content of the world model, and the observations it gets about the environment.

Later on, this model will be expanded into an SoS, and it will be shown how the content of the world model is affected at each of the three higher interoperability levels.

Technical interoperability: Transferability of data

The first level of LCIM deals with how to transfer data, and this problem can in turn be subdivided into several layers as done in the OSI model (ITU-T, 1994). Aspects included here are e.g. addressing in a network; ensuring delivery of messages; maintaining sessions; and ensuring security.

Although the semantic web technologies are not concerned with the basic transfer of data, it is worth noting that Internet technology is playing an increasingly important role in SoS. This is obvious in applications such as the Internet of Things (IoT), but there is also a more and more common scenario where a system is given a representation in the cloud through a private data link, and then interactions with other systems occur between their respective cloud representations (Axelsson, Fröberg, and Eriksson, 2019). This cloud-to-cloud interaction is also based on Internet and WWW protocols, such as HTTP. Since the semantic web was developed in the context of the WWW, these solutions for technical interoperability thus form an excellent solution for many SoS as well.

Example. For autonomous vehicles, several communication routes are possible. There will surely be vehicle-to-cloud communication of the kind mentioned above, using cellular networks. Also, there will be communication with the different mediators, and this can either be over the Internet directly from the vehicle, or through its cloud representation. Finally, in some situations there will be direct wireless vehicle-to-vehicle short-range communication using WLAN derivatives, for particularly local and time-sensitive communication.

Syntactic interoperability: Structure of data

The technical interoperability ensures that a data packet, consisting of a sequence of bits, can be securely delivered to its destination in its original representation. The next step is then to figure out what entities this sequence of bits represents. Traditionally, when data transfer was expensive, it was an art to squeeze in as much information as possible into these sequences, but nowadays when engineering costs are more dominant, flexibility and understandability of the data becomes more important, and therefore it is now common practice to represent data as text strings. This is also the approach of the semantic web, which provides several alternative formats including XML (W3C, 2014a) and JSON. However, there are also specific formats and we will in this paper use one called Turtle (W3C, 2014b) due to its superior readability.

As was mentioned earlier, RDF works with data in the form of triples to represent all kinds of 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 resource identifiers (also known as blank nodes). The blank nodes are useful for transient information where it is not necessary to express the identity of a resource but only what combination of properties it has. Since the same resource identifier may appear in several triples in K, this set can also 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.

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 fields are replaced by a wildcard. The result is the set of all triples matching the pattern. However, sometimes more advanced queries are needed that relate several resources to each other or add more constraints on relations. For this, the SPARQL query language has been developed (W3C, 2014c).

The triple stores often also have other useful features. One is the ability to create a conjunctive triple set out of separate subsets. This makes it possible to sometimes work on the entire union of all information, and at other times work with just a subset. For example, all the data related to a certain other system could be in one subset. Later on, this other system may lose its relevance to the first system (e.g., when an SoS constellation is dissolved), and then the whole subset can simply be deleted, without having to search in a larger set of triples to identify which ones should be removed.

Example. In the autonomous vehicle example, it is necessary to represent information about cars, and each individual car needs to be given an IRI. Today, all cars have a unique vehicle identification number (VIN), and this is one option for ensuring they have unambiguous names. Another option is to let each OEM give names to their cars in whatever way they like based on their web domain name. It doesn't matter which of these options are used, and they can be freely mixed, as long as a car knows its own IRI. We will assume that an OEM with the web address www.oem.com uses the second naming scheme, with <http://www.oem.com/car_123> being an example IRI for one of its cars.

To enhance readability, it is possible to introduce abbreviations for common IRI prefixes. For example, we could use "oem:" to be an abbreviation of <http://www.oem.com/>, and the car in the example can then be referred to as oem:car_123. This abbreviation can be seen as introducing a namespace for resource identifiers named by this OEM. We will also include a second namespace "ex:" as an abbreviation for <http://example.com/> that will be used to write general concepts of the SoS more succinctly.

The IRIs can now be used to express relations, such as the property that a vehicle has a certain color and speed, as illustrated in the following Turtle data[1]:

```
@prefix oem: <http://www.oem.com/> .     # Prefix declaration
@prefix ex: <http://example.com/> .
oem:car_123 ex:has_color "red" .         # car_123 has_color red
oem:car_123 ex:has_speed 25 .            # car_123 has_speed 25
```

The last line is a triple, consisting of the IRI of the car as a subject; the IRI of the property relation "has speed" as the predicate; and the literal 25 as the object.

To achieve syntactic interoperability using RDF, it is sufficient to decide what RDF textual formats are allowed, and then let each system include the parsers for those formats to give them the triples.

A very important aspect to notice is that the IRIs used are just names. Although they may contain some information, such as the name of the company, or a VIN that could have a meaning, this information should be ignored when working on the data, and any other valid IRI string can be used equally well as long as usage is consistent. If it is interesting to access the VIN, it should be included through a triple with a predicate "has VIN" and the actual identifier as a string.

Semantic interoperability: Meaning of data

On the semantic interoperability level, the meaning of data is agreed. An interpretation of this is that two systems, if given the same data, will draw the same conclusions. This requires some notion of reasoning, and a language where certain symbols have pre-defined meaning that is understood by the reasoning mechanism.

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

[1] In Turtle syntax, the line starting with "@prefix" declares a namespace. Triples are written on the format "subj pred obj .". Comments start with the "#" symbol and continues to the end of the line.

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 therefore been standardized and are universally accepted. Those constructs come from description logic, and include:

- *Classes and instances.* A resource identifier may refer to a class of objects, and it can be expressed that the class is the *type* of another resource (the class's instance).
- *Properties:* A resource identifier 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 (W3C, 2014d), which uses the namespaces "rdf:" and "rdfs:".

RDFS contains a small set of constructs which are very common. A more elaborate extension is in the Web Ontology Language version 2 (OWL2, with the namespace "owl:") that allows the expression of much richer information about resources, classes, properties, etc. (W3C, 2012). 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 identifiers 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.

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 (ISO, 2017) and the Friend of a Friend (FOAF) vocabulary defines terms related to people and their relations (Brickley and Miller, 2014). 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.

Since RDF is based on description logic, it is possible to perform reasoning on a set of triples, to see what implications it has. As an example, consider a superclass and a subclass, where a certain property is defined on the superclass. A reasoner can figure out that the subclass by implication also has this property and add the corresponding triple to the knowledge base. Although this does not change anything on a logical level, the practical implication is that all relations become explicit and can be found directly by a database search.

Example. 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 vehicles[2]:

[2] The syntax of Turtle allows the grouping of triples that share the same subject, so "subj pred_1 obj_1 ; pred_2 obj_2 ." is equivalent to the two triples "subj pred_1 obj_1 . subj pred_2 obj_2 ." Furthermore, Turtle uses the abbreviation "a" to mean the very commonly used predicate "rdf:type" which indicates that a resource is an instance of a class.

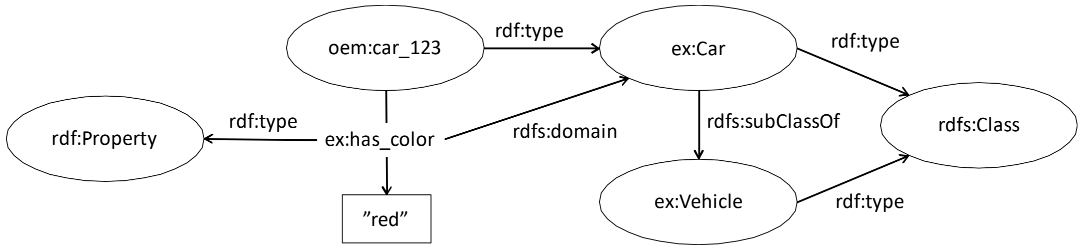

Figure 2. RDF linked data represented as a graph.

```
oem:car_123 a ex:Car .                    # car_123 is a Car
ex:Vehicle a rdfs:Class .                 # Vehicle is a class
ex:Car a rdfs:Class ;                     # Car is a class...
        rdfs:subClassOf ex:Vehicle .      # ...which is a subclass of Vehicle
ex:has_color a rdf:Property ;             # has_color is a Property...
              rdfs:domain ex:Vehicle .    # ... whose domain is Vehicle
```

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.

Figure 2 illustrates that it is also possible to represent the RDF information as a graph, where all IRIs and literals become nodes and the predicates are represented as links between them. The "meaning" at the semantic level is to a large extent represented by the patterns that occur in such graphs. Note that this meaning is provided through data, rather than being hard coded into the systems, which greatly enhances extensibility and evolution as well as reuse of standard software components.

Pragmatic interoperability: Use of data

We will now return to the basic system model introduced in Figure 1, and expand it into an SoS model, as illustrated in Figure 3. Here, two systems are included together with an environment, which is shared but where different entities are of interest to each of the systems. These also have their own states; functions; world models; and communication links between them. The technical, syntactic and semantic interoperability levels deal to a large extent with the communication and can be handled in an SoS using standard communication technology combined with RDF. However, on levels 4-6, it is necessary to dig deeper into the systems, and look at the content of the world model. In Figure 3, the world model of System 1 has been expanded to show that it can, in an SoS setting, contain not only information about its own environment E1 (on levels 0-3), but also potentially information about System 2's states S2 (on level 4), functions F2 (level 5), and world model WM2 (level 6).

So far, triples have been used to capture statements such as "car_123 is red". However, on the higher levels of interoperability, it is necessary to include also information such as "car_123 believes that 'car_456 is blue'". In this case, the object of the statement is a statement itself, and to handle this in RDF, a method called reification is used. The example statement can be represented as follows:

```
_:stmt1 a rdf:Statement ;                 # stmt1 is a statement (a triple) ...
           rdf:subject oem:car_456 ;      # ... whose subject is car_456
           rdf:predicate ex:has_color ;   # ... predicate is has_color
           rdf:object "blue" .            # ... object is "blue"
oem:car_123 ex:believes _:stmt1 .         # car_123 believes stmt1 is true
```

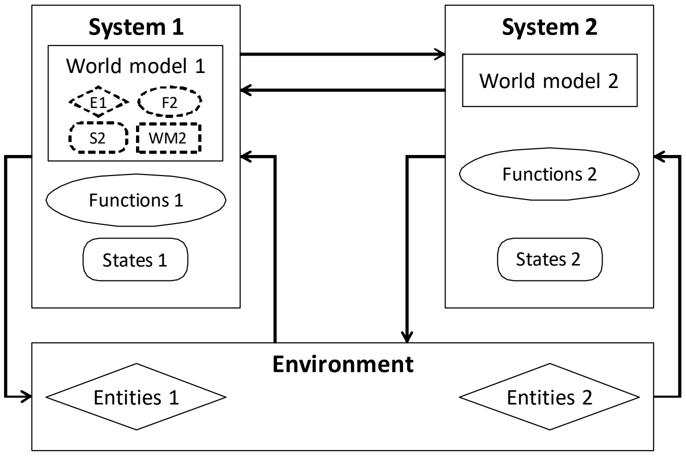

Figure 3. A basic system-of-systems model.

Here, ”_:stmt1” refers to a “blank node” naming a statement, since it is not interesting to give any specific identity to this resource. The node has the type rdf:Statement, which consists of a subject, predicate, and object. (Similar ideas as reification can also be used for other purposes where structured data needs to be represented, such as clarifying the SI unit of a literal number, which belongs to the semantic level.)

Statements can be qualified based on a number of aspects:

- Who holds the belief?
- Who made the statement originally?
- How certain is the statement?
- When was the statement assumed to be true (T)?
- When was the statement made (T’)? If T = T’, this means that it is a statement about the current state, whereas T < T’ represents historical data, and T > T’ represents forecasts.

On the pragmatic interoperability level, the world model of a system is primarily extended with state related information about other systems, and this often requires reification. The state related information includes:

- *Observations*. These can be observations of a part of the environment, observations about the system itself, or observations obtained through communication.
- *Modes*. This can include generic CS states such as being ignorant, prepared or passive (Axelsson, 2019a), but also application specific modes or current activities in a workflow.
- *Configurations*. Both static configurations, such as what communication protocols a system understands, and dynamic configurations, such as what connections it currently has to other CS as part of an SoS constellation, are of relevance.
- *Services*. Descriptions of the available functions and how to request them is needed.

Pragmatic interoperability requires that the systems can exchange information describing the services they offer and when they are available, and such constructs thus have to be included in the ontology.

Example. In autonomous driving, an example where pragmatic interoperability is needed is platooning, where one vehicle automatically follows another at a short distance, thereby forming a road train. This requires wireless communication between those vehicles to maintain a short yet safe distance between them. Here, a vehicle could be in a state where it is willing to become a platoon leader, and if that is the case, another vehicle can request to join it as a follower. Once in the platoon, the vehicles may perform various operations that require mutual understanding of states, such as dissolving the platoon or performing a lane change.

Dynamic interoperability: Effects of data

The dynamic interoperability mainly requires the inclusion of information related to functions of other systems in a system's world model, such as:

- *Missions*. This includes what objectives a CS is pursuing, what workflows it is using to accomplish this, and what constraints it has.
- *Capabilities*. Understanding what another system is capable of is necessary to plan collaborations and knowing what can be expected if a task of one's mission is assigned to another system. This includes not only capabilities in general, but also what is possible at a specific time (depending on what other tasks that system is already carrying out).
- *Properties*. The characteristics of different functions are often controlled by various properties. These can include calibration parameters of a control system, but also physical properties such as weights. Having information about such properties in a related system greatly improves the ability to predict how it will react to different stimuli.

To achieve dynamic interoperability, the ontology has to capture recurring concepts related to missions, capabilities, and properties of function dynamics. The later include both internal state changes and updates to the environment. An initial attempt in this direction is described by Axelsson (2019b).

Example. In a platoon of autonomous vehicles, the lead vehicle may need to perform an emergency brake to avoid a hazardous situation, and this requires that all follower vehicles also brake urgently. However, there can be variations in brake performance between the vehicles, e.g., depending on their current weight, and this means that an emergency brake to avoid a collision with someone outside the platoon may in fact trigger a collision inside the platoon. If the vehicles exchange information about their ability to perform the brake function, brake power can be adjusted, or a different strategy than emergency braking can be chosen to avoid the hazard. Another example is when the mission of the lead vehicle in the platoon requires it to leave a highway at an exit, which means that the second vehicle may need to step in as platoon leader. Having access to this information well in advance allows the change of responsibility to be planned at an appropriate time.

Conceptual interoperability: Mutual awareness of data

The conceptual interoperability in general also goes in the direction of more advanced and commonly agreed ontologies, but it further includes the mutual awareness of the knowledge that the involved systems possess, and a self-reflective capability. This means that System 1 needs to have in its world model a representation of System 2's world model (and vice versa).

This recursive relation may make one think that now everyone has to know everything about everyone else, and that a common global information set is required. This is however not at all the case, since each system chooses what it should include in its own world model. The world model is, as the name implies, merely a model and thus a simplified representation of reality. Although all information about the states, functions and world model of another related system are candidates for inclusion in the ego world model, System 1 only needs to include in its world model as much information about System 2's world model as it has use for. The law of parsimony ("Occam's razor") thus applies, and the world model should not be more detailed than is necessary. At the same time, the law of requisite variety (Ashby, 1956) is also valid, which implies that the world model has to be as detailed as the level of control the system wants to achieve in its environment. There is thus a delicate balance to strike.

The conceptual model does not have to be monolithic and known in every detail within an SoS, but different parts of the SoS can have different versions of it that add more detail which is specific to a certain role or task. In most situations, there are however some core elements that are common and that can be subclassed to add more details. In this way, a certain basic language is shared. This flexibility of conceptual models is easily handled in RDF, and a strength is that the conceptual model is just data (a set of triples). Thereby, systems can ask each other about the vocabulary and concepts they are using and exchange that meta-information as well without additional mechanisms.

It is worth noting that whereas the states and functions of a system are sometimes visible on the outside, and sometimes not, the world model is always hidden and not observable. This means that the primary means of obtaining information about it is through communication, although this can sometimes be complemented with sensor observations combined with reasoning.

Example. In the autonomous vehicle example, a certain car may be planning to change lane, but there is a truck in the new lane at a hazardously short distance. If that truck is aware of the imminent lane change, it can be expected that it will adjust its speed to make room, but if it is unaware, there is a large risk of an accident. Thus, if the car knows that the truck knows about the car's plans, it can choose a different course of action than in the opposite case.

Discussion

We have applied RDF as a basis for SoS interoperability in some larger cases (Axelsson, Fröberg, and Eriksson 2019; Axelsson 2019b), and some of the experiences will now be described.

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 (Oracle, 2016), even this should usually not be an issue. If the cloud-based architecture discussed earlier is used, performance is in particular not an issue. However, for very fast control signals, such as directly between the autonomous cars in the example, RDF may not be the best solution. It also has less value in those situations, since they tend to involve a small and fixed data set which does not require the expressiveness and flexibility of linked data.

The world model used on the higher levels of interoperability has some challenges related to it. It can sometimes be difficult so select what to include, and for how long data should be saved. There is an apparent risk of collecting excessive information, and at some point this will have cost penalties in terms of lookup performance or communication overhead.

Getting access to internal data of another system can also be difficult, at least in commercial competitive situations such as collaborative SoS. Internal data can have a value, and an independent CS may not be willing to share it with others unless it receives some compensation for doing so. There are also issues with trust, and it cannot always be taken for granted that the data provided is correct and accurate. If there is doubt, contractual mechanisms may be needed but this leads to a lot of complexity. Finally, some systems which deal with personal information may not be allowed to share data freely due to privacy regulations. Trustworthiness in SoS is discussed more by Axelsson (2020).

The world model idea used in this paper is somewhat similar to the concept of "digital twins" which is commonly discussed in initiatives such as Industry 4.0. A digital twin allows a shared representation of physical assets by combining engineering data with runtime data. It thus provides a model of the current state of a system and based on this model virtual experiments can be performed to evaluate the effects of alternative courses of actions. However, in the context of SoS, the digital twins of the CS will partially overlap, and there is thus a need for continuous alignment. We believe that RDF can be a suitable framework also for representing and communicating digital twin data.

Related work

LCIM is one of several interoperability maturity models, and it is compared to some alternatives by Guédria et al. (2008), including LISI, OIM, and EIMM. The comparison indicates that LCIM is more technically oriented whereas some of the others emphasize management and organizational aspects to a higher degree. This is a weakness of LCIM in general, but since the purpose of this paper is to relate a technical framework (the semantic web) to interoperability, it is actually a strength.

Although LCIM has its origins in the simulation domain, it has been used in various SoS-like settings as well. Kolbe et al. (2017) discuss its usage in the context of an open IoT ecosystem and suggest solutions at each of the interoperability levels. Panetto and Molina (2008) apply it to integration in manufacturing systems and compare it to some other interoperability maturity models.

Abdalla et al. (2015) 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, in order to ease integration, engineering activities, as well as SoS management.

The use of linked data for SoS interoperability has been suggested in a number of papers, including Axelsson, Fröberg, and Eriksson (2019) and Axelsson (2019b) which apply it in the construction domain. Baek et al. (2018) present the M2SoS meta-model for SoS, which was elicited in the context of a mass causality incident response system. Zhu et al. (2017) propose an OWL2 based ontology for describing SoS missions and discuss principles for how the mission may be decomposed and allocated to CS. 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. Within Industry 4.0, RDF has also been suggested as a means of achieving interoperability (Bedenbender et al., 2016; Grangel-Gonzalez et al., 2017).

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 organization (Saadatmand and Bucaioni, 2014).

Conclusions

Interoperability is generally accepted as one of the cornerstones of SoSE, and there is a need to develop more systematic and efficient engineering methods for finding the right level of interoperability. This is a trade-off between functionality and lifecycle cost, as are many aspects of systems engineering. LCIM provides a useful model for analyzing and describing the level of interoperability that is appropriate in a certain situation, and RDF is a powerful technical framework that can resolve many interoperability challenges. By combining the two, we believe that systems engineers are better supported in finding the right solution for their particular SoS problem.

Interoperability is often thought of as how well different systems can understand each other. However, to achieve that understanding it must be ensured that the engineers behind the systems have a common view of the terminology to use, because it is their encoding of this knowledge in the systems that makes the systems understand. RDF has the benefit that the specifications are both accessible to humans and to machines, and this makes it possible to encode the mechanisms for understanding at a higher level in the systems, using standard techniques and providing flexibility for SoS evolution.

Beyond this, there are many areas where future work would be fruitful. In particular, the higher levels of interoperability require the development of upper ontologies that provide a conceptual foundation. Some elements of this is domain and problem specific, but most likely there are also recurring SoS concepts, such as missions, tasks, capabilities, constraints, constellations, etc. Understanding these even further, and providing a foundational ontology for them, would contribute significantly to making SoSE more efficient and robust.

Although LCIM is a very useful model for reasoning about interoperability, it lacks formal rigor which sometimes makes it difficult to interpret. Therefore, a formalization of the levels of interoperability would also be a useful continuation of this work.

If RDF is used for operational data as well as exchange of engineering data through OSLC, this provides a strong foundation for the establishment of distributed digital twins for SoS. This could potentially allow unprecedented support for analysis and evolution, but more research is needed on how to organize such combined runtime and design models to make them practical.

References

Abdalla, G., *et al*. 2015, ‘A Systematic Literature Review on Knowledge Representation Approaches for Systems-of-Systems’, in *IX Brazilian Symposium on Components, Architectures and Reuse Software*, Belo Horizonte.

Axelsson, J. 2015, ‘A Systematic Mapping of the Research Literature on System-of-Systems Engineering’, in *IEEE Systems of Systems Engineering Conference*, San Antonio, Texas.

——— 2019a, ‘A Refined Terminology on System-of-Systems Substructure and Constituent System States’, in *IEEE Systems of Systems Engineering Conference*, Anchorage, Alaska.

——— 2019b, ‘Experiences of Using Linked Data and Ontologies for Operational Data Sharing in Systems-of-Systems’, in *IEEE Systems Conference*, Orlando, Florida.

——— 2020, ‘A Unified Approach to System-of-Systems Trustworthiness Analysis Based on Systems Theory’, in *INCOSE Intl. Symposium*, Capetown, South Africa.

Axelsson, J., Fröberg, J., and Eriksson, P. 2019, ‘Architecting systems-of-systems and their constituents: A case study applying Industry 4.0 in the construction domain’, in *Systems Engineering*, 11(6), pp. 455-470. 10.1002/sys.21516.

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

Baek, Y. *et al.* 2018, ‘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, pp. 1–7.

Bedenbender, H. *et al.* 2016, *Examples of the Asset Administration Shell for Industrie 4.0 Components – Basic Part*. ZVEI - German Electrical and Electronic Manufacturers' Association.

Berners-Lee, T., Hendler, J. and Lassila, O. 2001, ‘The Semantic Web’, *Scientific American*, 284(5), pp. 28–37. doi: 10.2307/26059207.

Bianchi, T., Santos, D. S. and Felizardo, K. R. 2015, ‘Quality Attributes of Systems-of-Systems: A Systematic Literature Review’, in *2015 IEEE/ACM 3rd International Workshop on Software Engineering for Systems-of-Systems*. IEEE, pp. 23–30. doi: 10.1109/SESoS.2015.12.

Brickley, D. and Miller, L. 2014, ‘FOAF Vocabulary Specification’, http://xmlns.com/foaf/spec.

Dahmann, J. 2014, ‘System of Systems Pain Points’, in *INCOSE International Symposium*, pp. 108–121. doi: 10.1002/j.2334-5837.2014.tb03138.x.

Grangel-Gonzalez, I. *et al.* 2017, ‘The industry 4.0 standards landscape from a semantic integration perspective’, in *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, pp. 1–8. doi: 10.1109/ETFA.2017.8247584.

Guédria, W., Naudet, Y. and Chen, D. 2008, ‘Interoperability maturity models–survey and comparison’, in *OTM Confederated International Conferences,* pp. 273-282, Springer.

IEEE 1990, ‘IEEE SA - 610.12-1990 - IEEE Standard Glossary of Software Engineering Terminology’.

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

ITU-T 1994, *Information Technology - Open Systems Interconnection - Basic Reference Model: The Basic Model,* Recommendation X.200.

Klein, J. and van Vliet, H. 2013, ‘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, New York, USA: ACM Press, p. 13. doi: 10.1145/2465478.2465490.

Kolbe, N., Robert, J., Kubler, S. and Le Traon, Y. 2017, ‘PROFICIENT: Productivity Tool for Semantic Interoperability in an Open IoT Ecosystem’, *Proc. 14th Intl. Conf. on Mobile and Ubiquitous Systems*, pp. 116-125, Melbourne, Australia.

Oracle 2016, ‘Oracle Spatial and Graph: Benchmarking a Trillion Edges RDF Graph’, White paper.

Panetto, H. and Molina, A. 2008, ‘Enterprise integration and interoperability in manufacturing systems: Trends and issues’, *Computers in Industry*, 59:641-646.

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

Tolk, A., Turnitsa, C. D. and Diallo, S. Y 2006, ‘Ontological Implications of the Levels of Conceptual Interoperability Model’, in *World Multi-Conference on Systematics, Cybernetics and Informatics*, Orlando.

Wang, W., Tolk, A. and Wang, W. 2009, ‘The Levels of Conceptual Interoperability Model: Applying Systems Engineering Principles to M&S’, in *Spring Simulation Multiconference*.

World Wide Web Consortium (W3C) 2012, ‘OWL 2 Web Ontology Language Quick Reference Guide’, 2nd Edition. URL: https://www.w3.org/TR/owl-quick-reference/.

——— 2014, ‘RDF 1.1 Primer’. URL: https://www.w3.org/TR/rdf11-primer/.

——— 2014a, ‘RDF 1.1 XML Syntax’. URL: https://www.w3.org/TR/rdf-syntax-grammar/.

——— 2014b, ‘RDF Turtle 1.1: Terse RDF Triple Language’. URL: https://www.w3.org/TR/turtle/.

——— 2014c, ‘SPARQL 1.1 Overview’. URL: https://www.w3.org/TR/sparql11-overview/.

——— 2014d, ‘RDF Schema 1.1’. URL: https://www.w3.org/TR/rdf-schema/.

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

Biography

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 in the automotive industry with Volvo Group and Volvo Cars 1997-2010. He is now a full professor of computer science at Mälardalen University, Sweden and a senior research leader in systems-of-systems at RISE Research Institutes of Sweden. His research interests are focused on all aspects of systems-of-systems engineering, and in particular system architecture. Prof. Axelsson is a member of INCOSE and has served as chairman of the Swedish chapter.