34th Annual INCOSE international symposium
hybrid event
Dublin, Ireland
July 2 - 6, 2024

An Agent-Based Ontology to Support Modeling of Socio-Technical Systems-of-Systems

Jakob Axelsson
Mälardalen University and RISE Research Institutes of Sweden
PO Box 883, SE-721 23 Västerås, Sweden
jakob.axelsson@mdu.se

Abstract. Systems-of-systems are characterized by the independence of their constituent elements. Such an element is usually socio-technical, comprising technology, humans, or organizations. To capture its independence, it needs to be viewed as an intelligent agent that relies on an internal model of the world for its decision-making. Hence, a system-of-systems model will include multiple agents that inside themselves contain different models of the same system-of-systems. Describing these overlapping subjective models and their usage by the agents is essential to properly understand the resulting behavior of the overall system-of-systems. Current modeling practices are not well suited for dealing with this, and the paper therefore outlines an ontology that makes the agents and their internal models more explicit. The paper also discusses the implications such models have on systems engineering practices and how they address known system-of-systems engineering pain points.

Keywords. Systems-of-systems, modeling, analysis, agents, ontology.

Introduction

A *system-of-systems (SoS)* is a situation where the elements of a larger system are themselves complex entities with their proper objectives. An SoS element is called a *constituent system (CS),* and it is characterized by having both operational and managerial independence vis-à-vis the SoS (Maier 1996). The CS of an SoS choose to interact by exchanging information to create capabilities they cannot achieve on their own (ISO/IEC/IEEE, 2019). The CS are normally *socio-technical*, where the social elements provide managerial independence that allows the SoS to evolve continuously by adapting to circumstances.

It has long been understood that SoS engineering (SoSE) has particular challenges or "pain points" that set them apart from SE for integrated systems (Dahmann 2014). Underlying these challenges is a lack of proper nomenclature for describing SoS. However, as Baxter & Sommerville (2011) argue, current modeling techniques are not adequate for socio-technical systems. Given the socio-technical nature of SoS and the key role played by CS independence, it is essential to improve SoS modeling practices to deal explicitly with these concerns.

The nature of CS independence is not well articulated in much of the work on SoS, but it must lie in the ability of CS to make decisions on their own (Axelsson & Svenson, 2022). These decisions are based on their subjective understanding of value, enabling them to evaluate and choose individually among alternative courses of action. A reasonable hypothesis is that making the nature of CS independence explicit in SoS models will open roads for dealing with several of the pain points and socio-technical challenges.

In this paper, I will use the term *agent* as an abstract concept that describes an entity with agency, i.e., that can take actions and choose what actions to take. The ability to choose actions means that agents need to contain advanced internal world models, allowing them to exhibit the value-driven decision-making ability that is at the core of being independent. In situations where several agents are present and collaborating, there is also a need for them to exchange model information, which puts requirements on their ability to communicate and understand each other.

The abstract agent concept can be applied to various concrete entities that exhibit agency, such as humans, technical systems, and organizations. It can also be applied to the CS of an SoS, and hence be used to capture the nature of CS independence. I use the abstract term agent when the concrete nature is not essential to the discussion, and use more concrete terms when discussing specific aspects that do not apply to agents in general.

The contribution of this essay is to suggest a conceptualization that covers the nature of independent agents and to analyze some of the implications it has. This turns out to require a deeper understanding of what models and systems are. Such an ontology will be a useful basis both for addressing the SoS pain points through research as well as aiding practitioners who need to find appropriate models of concrete SoS situations here and now.

The remainder of the paper is structured as follows. First, some fundamental concepts in systems theory and modeling are revisited, that are essential for SoS applications. Then, the suggested ontology is introduced, starting with a foundation that covers general systems, followed by an extension elaborating on agent concepts. The implications for practitioners and researchers are subsequently discussed, together with some remarks on how the ontology addresses the SoS pain points. Finally, the conclusions are summarized.

Models and Systems

The notion of a *system* traces its roots back to Aristotle's *Metaphysics* (Section 8.6), stating that the whole is other than the parts. The essence is that a system can be seen as a set of interrelated parts, and the interaction between the parts gives rise to properties of the whole. As simple as this is, many deep questions hide underneath the surface. In this section, I will explore a few of these questions since they give an important foundation for the ontology to be presented later.

System Worldviews

The basic definition of a system allows several different interpretations. The system worldviews held by SE practitioners are discussed in detail by Dori et al. (2019). Some of these, such as the "constructivist" and "mode of description" views, consider systems to appear in a model of the mosaic of the real world, rather than being something that physically exists. It is thus in the mind of an observer that systems appear, as elements in a cognitive model.

This is not a question of whether a physical reality exists or not (I assume it does), but rather that there are no uniquely given boundaries between parts of reality. This means that if one wants to describe the world as systems composed of parts, there are choices to be made on exactly what elements of reality correspond to those systems and parts.

This constructivist worldview is well-established in soft systems (Checkland, 1993; Reynolds & Holwell, 2010, p. 7), where it is essential to capture that socio-technical agents view the world differently. They act based on their world models, and hence understanding their actions requires understanding their world models. Since an SoS consists of CS that retain some independence, an SoS displays important characteristics of soft systems, and the differences in world models among agents are equally important here. Therefore, the constructivist worldview is adopted throughout this paper.

Characteristics of Models

The worldview that systems exist in a description inside an agent, not in the real world, brings the notion of *models* to the foreground. Minsky (1965) defines models as follows: "To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A." Stachowiach (1973), in his General Model Theory, gave three essential characteristics of models (see also Figure 1):

- *Mapping*: A model is a representation of some original, which could be natural or artificial.
- *Reduction*: A model does not include all properties of the original, but it is a simplification.
- *Pragmatism*: The model is intended to work as a replacement for the original, with a particular purpose for its user.

Apostel (1960) makes the presence of the observer even more explicit, stating that a model should be understood from its relationship to its user, its purpose, and the original. Rothenberg (1989) mentions cost-effectiveness as a model characteristic. Essentially, this is a restatement of the principle of parsimony, or "Occam's razor."

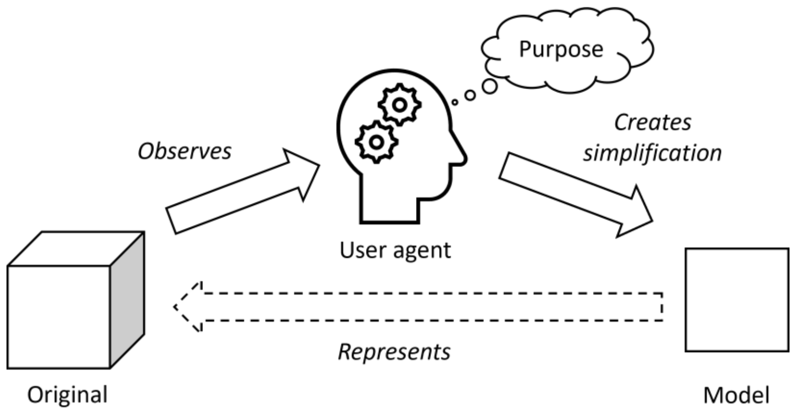

Figure 1. Key elements of modeling, based on Stachowiach (1973) and Apostel (1960).

As famously pointed out by Box (1976), “all models are wrong, but some are useful.” If two models always lead to the same answers to the questions of interest, parsimony requires that the simpler one is chosen. However, if the purpose is to choose among alternative actions, there is also a lower limit on how simple the model could be. As stated in Ashby’s (1956) first law of cybernetics, the model must contain at least as much information as the number of relevant actions the user agent can choose between.

Several of the sources above use the term “observer” to denote the agent that creates the model. This highlights that the agent uses its perception of the original in creating its simplified representation. However, it does not mean that a model can only contain observable elements. On the contrary, many models include hypothetical elements that cannot be observed, but adding them can simplify the model without reducing its explanatory power. The history of science is full of examples where such unobserved elements were introduced into scientific laws, and only much later validated when new measurement techniques appeared. Another case is in the engineering of new products. These products cannot be observed since they do not yet exist, but their models can still be constructed by agents. The purpose is not to explain observations, but rather to predict properties and guide product development. Apostel (1960) discusses no less than ten different modeling use cases in empirical sciences, and most of these are highly relevant in engineering too.

Basic Ontology for System Models

Having established that systems can be seen as entities in a model created by a certain agent, it is necessary to detail some other types of entities that are present too. The INCOSE systems definition (Sillitto et al., 2019) contains several such concepts (that I have marked with underlining): ”an arrangement of parts or elements that together exhibit behavior or meaning that the individual constituents do not. [...] The system’s properties (as a whole) result, or emerge from: the parts or elements and their individual properties; AND the relationships and interactions between and among the parts, the system, and its environment”.

In this section, an ontology is introduced that covers these commonly agreed-upon systems concepts. An ontology is here understood as a list of concepts defined in terms of each other. For an introduction to ontology development, as well as foundational concepts of the systems domain, see Rousseau et al. (2018). The essential concepts of the proposed systems ontology and their interrelations are illustrated informally in Figure 2 and are elaborated upon in the remainder of this section.

Structure

A *model* is a set of *elements* and *relations*. Relations exist between pairs of elements, and they are in general directed so that one element may be related to another but not necessarily vice versa.

An element can have different *properties*, that are mappings from elements to some value spaces. Since relations are directed, they can be represented as special properties of an element indicating which other elements it is related to.

A special type of relation is *composition* which indicates that one element is a *part* of another element, where the larger element is called a *system*. An element may be part of several systems simultaneously, such as a CS having roles in multiple SoS.

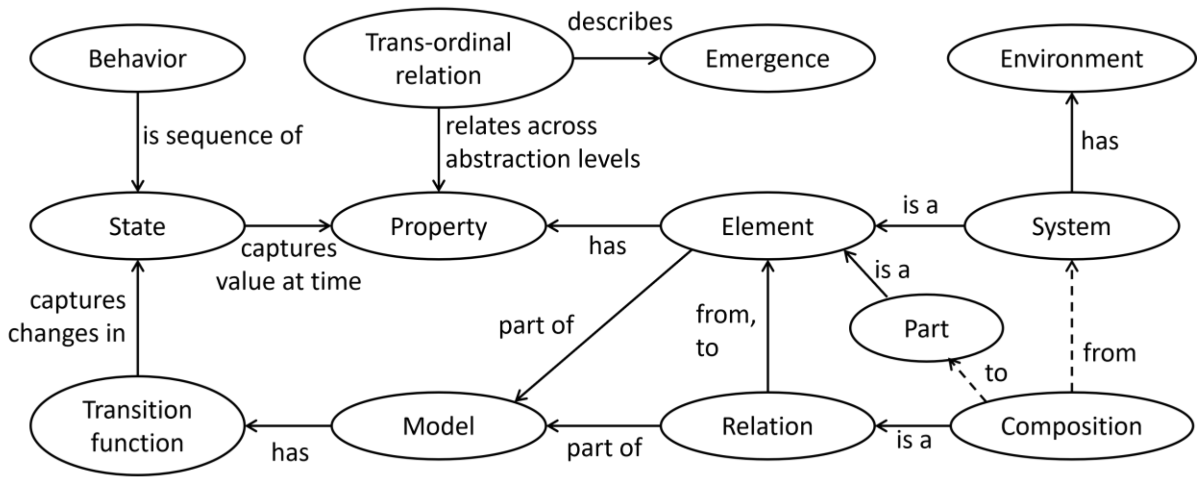

Figure 2. Informal and partial representation of key concepts in the basic system ontology.

A system can exist in an *environment*, which is all the elements in the model that are not parts, directly or indirectly, of that system.

Behavior

At any point in time, each property of an element takes on a certain value, and an element's *state* is captured by its properties at that time. The whole model's state is the combination of all elements' states.

By regarding relations as element properties, the relations also become dynamic and can be seen as parts of the model state. This applies equally to composition relations between systems and parts. For instance, a CS may join or leave an SoS.

The state of the model changes over time and the transition between states is captured in its *transition function*. The term *behavior* denotes a sequence of state transitions of a model or an element. The transition function is defined on model states, and not on individual element states. Therefore, it is possible that the new state of an element not only depends on the prior state of that same element but also on the states of other elements. However, this can only happen if there is a relation between the elements, which captures that behavior depends on both individual elements and their relations.

Trans-ordinal Relations and Emergence

A system is, as explained above, composed of several interrelated parts. Therefore, it is possible to describe characteristics on the system level (capturing the nature of the whole) or on the part level (capturing the nature of the parts and their relations). Since a description on the system level will not contain a description of the parts, it has fewer details than a description on the part level, and can thus be seen as a description on a higher level of abstraction. Typically other properties are more relevant to the system than to the parts.

At the same time, the system and the set of parts are just two model representations of the same thing, and hence there should be some correspondence between behaviors expressed at the two levels. This correspondence is called *trans-ordinal* since it spans orders of abstraction. The properties and behavior of the system are thus *emergent* from the properties and behavior of the parts (Axelsson, 2022). The transition function of the system is related to the transition functions of its parts and the relations between them. The system behavior may depend on the parts' behaviors (upward causation) or the parts may depend on the system (downward causation).

Remarks on the Basic System Ontology

Before turning to the core part of this paper, namely the agent ontology, a few final remarks are needed about the choices made in the basic ontology. The starting point was INCOSE's definition of systems. However, there are several ways in which that definition can be interpreted.

One choice was the directional and dynamic nature of relations. This is not so emphasized in models of purely technical systems such as integrated products. Often, a physical relation is naturally bidirectional, and an integrated system mostly contains static relations. However, for socio-technical systems, a more general formulation is essential. One person may be able to observe another person, but not vice versa, and personal acquaintances come and go.

The description of the ontology above does not put requirements on model qualities, e.g., coherence, completeness, or soundness. Most definitions of the system concept would have such requirements, such as stating that all the parts of the system should be directly or indirectly related to each other (e.g., Ackoff, 1971). It is easy to agree on this, but I still choose to let the modeler decide on this issue of well-formedness. Ultimately, anything can be included or excluded from the model description as long as the purpose is fulfilled.

A model does not necessarily have to be explicit on how the emergence is created. As always, it is up to the modeler whether it suits the current purpose better to describe the behavior as relations between properties at the system level, or to describe the parts level relations and then define the trans-ordinal mapping from part properties to system properties.

Extending the Basic Ontology with Agents

The ontology introduced above provides a general language for describing systems. In this section, it is extended with concepts that describe elements having agency. Such agents can to some extent control their behavior by choosing what actions to take. The notion of agents used here has similarities with how they are defined in the field of agent-based modeling and simulation (see, e.g., Macal & North, 2005). However, I use concepts more closely aligned with common terminology in the systems and SE fields when defining the internals of agents.

The abstract concept of agents can model many types of concrete entities, such as humans, automated technical systems, and organizations. The reader should bear in mind that the concrete elements modeled as agents are not necessarily structured as I will suggest in reality. As with all models, this is a simplified representation that hopefully contributes to a certain purpose. The purpose for including agents in this paper is that I have found the concept essential when modeling SoS applications in various domains since it captures the independent nature of CS.

Agents

Agents are considered a type of element in the system ontology, and hence they have properties, states, relations, and behavior. The main aspect that distinguishes agents from other model elements is that they can choose their actions. To capture this, an agent needs a representation of the world around it. The agent uses this representation to reason about what actions to take, which makes them independent. An overview of an agent's internal structure is given in Figure 3, and the remainder of this section will provide more details about the various elements and relations in it.

World Model and its Underlying Ontology

At the heart of the agent is its *world model* which contains the information the agent keeps about its environment and about itself. The world model is derived from *observations* and is used to decide *actions*. It is thus a foundation for the agent's situation awareness.

The world model is a model exactly in the sense described previously: it is created by a particular agent; it represents some external entity; it is derived based on observations; it has a purpose; it is a simplification of the represented entity; and it can be expressed using the language of the basic *ontology* or some extension of it. Since the agent can interact with other agents, it needs to have models of those agents to reason about their actions and potential interaction.

Methods and Value

The purpose of the agent's world model is to decide on what actions to perform. This decision-making requires two additional elements, namely a notion of *value* and a decision-making *method*.

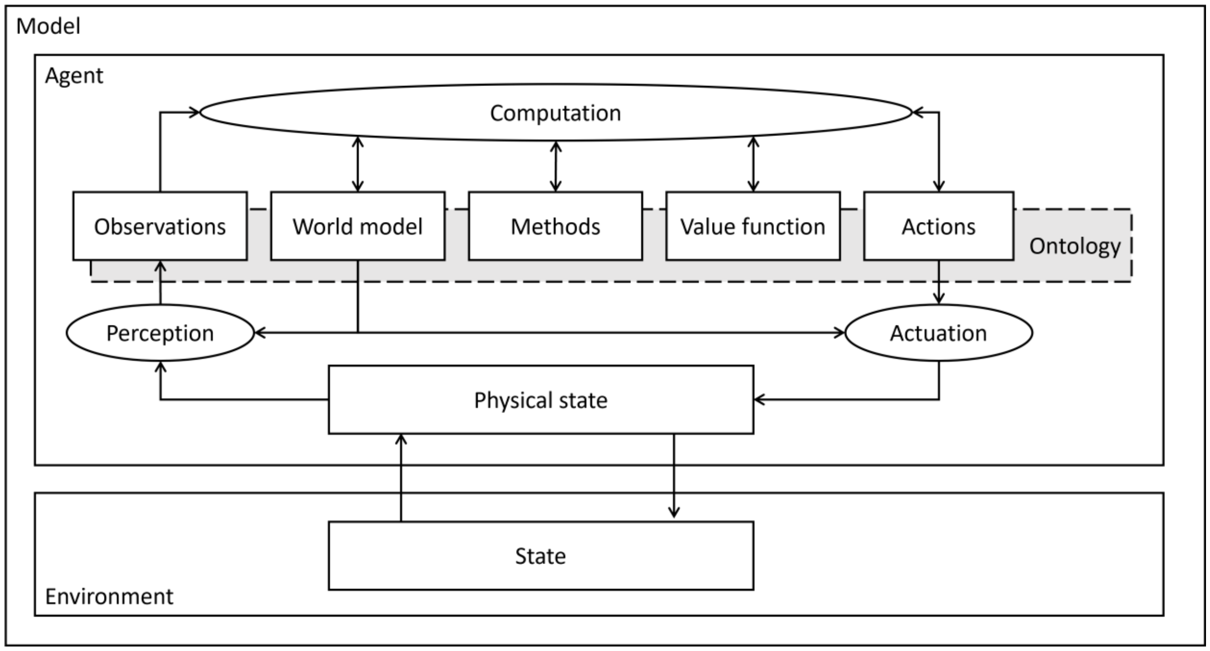

Figure 3. Structure of agent model elements.

The value function evaluates different states of the world that can result from various actions. It should be noted that those states need to be observable since the agent cannot possibly see a value in something it is unaware of.

The methods can be quite simple, such as feedback control that just decides on the next action based on observations of the current state. However, the agent could also look far into the future, and this requires deriving a plan of action that can be the basis for feed-forward control. The planning will involve an evaluation of many alternative behaviors of the agent and require simulation methods running faster than real-time to compare them. This would make the agent an anticipatory system as defined by Rosen (Louie, 2010).

The value function and method are properties in the world model, rather than fixed functions, to allow the agent to learn. It can thus adapt both its value function and its methods over time, due to changing circumstances in the environment and for self-improvement. This adaptative ability can be seen as a higher-level capability of the agent (Axelsson & Eriksson, 2023).

Computation

To decide what actions to take, the agent needs a reasoning capability. I will view this as a general *computation* capability, which takes as input all the information held by the agent, including observations, the world model, previous actions taken, as well as method and value function. The outputs are what actions to take next, and what updates are needed in the world model as part of the agent's learning.

The view that the agent contains a general computation capability has important implications. One such implication is that agents are physical. This is a consequence of Landauer's principle (Landauer, 1961) which states that all computation requires energy, and hence must be physical. It also implies that software on its own cannot be an agent, since the software is just instructions to a physical hardware that performs the computations.

In many cases, there will be limited time for the agent to make decisions. This is because the world changes while decisions are made and if the agent acts too late, it will not get the intended value. The available physical power limits what computation the agent can perform within the available time, hence the agent may exhibit bounded rationality (Simon, 1956) in its decision-making.

Models are always finite since they are created by agents with limited actions and computational capabilities. Functions that are infinite mappings, such as transition functions of an infinite state space, must have a finite representation using a mathematical formula or a computer program. In general, the agent's computation capability must be Turing equivalent. Since the method is part of the agent's state and can change over time, it can be thought of as the agent's equivalent of "software" that operates on the "data" in the world model.

Physical State and Communication

Since the agent is physical, it has a physical state. This physical state can interact with the states of other entities in the agent's environment. The world model of the agent contains information, which must also have a physical representation. However, it is considered internal to the agent and not directly accessible to other elements. Therefore, it is separated from the physical state and the exact mechanisms for its physical information representation are abstracted away from this model.

This mimics Minsky's (1965) observation that it is sometimes difficult to exactly identify where a model resides.

A consequence of hiding the world model is that one agent can never directly observe the internal world model of another agent. To share information about one's world model, communication is needed. This is an essential part of SoS since the geographical distribution of CS means that they are mainly related through information exchange (Maier, 1996). Communication can be modeled in the ontology using the Shannon-Weaver approach (Shannon, 1948): The sending agent actuates its observable physical state to encode a message. These state changes influence the physical state of another agent (possibly indirectly, through some channel). The recipient agent perceives, or decodes, those state changes resulting in observations.

However, for communication to work, additional information is needed in the world models to capture semantics, pragmatics, and other higher-order concerns of interoperability (Axelsson, 2020). These are essential issues in an SoS and thus fundamental to represent in the model of CS. It is insufficient to think about communication as arbitrary state changes observable by others. The sending agent needs to have a purpose of communication, in the sense of an intended effect on the recipient agent. According to basic cybernetics (Ashby, 1956), this requires the sender to have a model of the recipient, which the sender uses to determine how to bring forward the message.

Perception and Actuation

To bridge the gap between the agent's physical state and its internal world model, two additional functions are needed. *Perception* maps the physical state to *observations*, and *actuation* changes the physical state based on the *actions* decided by the agent. Since these functions act as translations between the physical world and the model representation, they need to be aware of the ontological concepts underlying the world model, and the observations and actions need to be expressed in terms of those concepts.

Perception is based on the agent's own physical state, which means it can potentially also observe its own "health". For instance, an agent may be in a deteriorated physical state that affects its capabilities. Being able to observe this may prompt the agent to try to restore the health before proceeding to other objectives. This self-consciousness is essential when planning a complex series of actions, which requires that the agent understands its capabilities and what effects they can generate.

Remarks on the Agent Ontology

Before proceeding to a discussion on the practical uses of this ontology, some final remarks are necessary about the agent representation.

It is important to emphasize the general nature of agents. I do not make any specific assumption as to whether they represent humans or technology, but the notion allows both possibilities. An agent can also be a system in itself whose elements are other agents, as it would be in a socio-technical organization consisting of humans and technology. In that case, the behavior of the agent system is emergent from the constituent agents.

An agent is an element in the basic ontology, and it can be viewed from the outside as a "black box" with a transition function. However, an atomic transition function would in many situations be very hard to predict and explain, which is why it makes sense to view it as a composition of some internal functions and a hidden information state. In this model, the transition function is (hypothetically) broken down into a composition of computation, perception, and actuation, to better explain the behavior.

The inclusion of computation in the agent model relates to Crutchfield's (1994) work on emergence. He discusses system complexity in relation to different classes of computational complexity. This provides a basis for classifying models and agents which could potentially give a more stringent understanding of the complexity of SoS and the appropriate models for reasoning about them.

It is interesting to note some parallels between the agent ontology and well-established typologies of knowledge, that again go back to Aristotle. The world model corresponds to *episteme*, that is knowledge about the world. The methods relate to *techne* in being procedural knowledge or skills. The value function, finally, can be thought of as a part of *phronesis*, or wisdom.

Ontologies and Models in Systems Engineering Practice

The ontology presented in this paper is intended as a foundation for building models that support SE and in particular SoSE, and I will now discuss its relation to SE practices.

A first usage is quite informal. Defining a set of relevant concepts for SoSE modeling creates a guideline for what information should be captured. Some questions an analyst should be asking when confronted with a complex socio-technical SoS situation are directly derivable from the concepts in the ontology: What CS exist (the agent concept in the ontology)? What are the characteristics of the CS (element properties, states, relations in the ontology)? How can they change over time (transition functions)? What actions can a CS take (actuation)? What information does it have about the world (observations, world model)? What objectives does it have (value function)? What information do CS need to exchange to be interoperable (differences in agent ontologies)? These questions can be used for a structured approach to deriving, e.g., a Concept of Operations (ConOps) document, and for discussions among stakeholders (who are themselves agents).

The ontology can be refined into a formal metamodel for implementation in tools for model-based engineering, as proposed by Yang et al. (2019), and Lu et al. (2022). The suggested agent ontology highlights the need for including socio-technical features that are not captured efficiently in contemporary modeling languages such as SysML. This is essential in SoSE, but also helpful in SE in general. Making the modeling agent and its purpose explicit supplies valuable information about the scope of the model and what kind of questions it was designed to answer. It can also capture important properties of the engineers that carry out SE (Axelsson, 2002).

A system model is the basis for analysis, so having appropriate ways of modeling SoSE is essential for effective design decisions. The distinguishing factor between SoS and other systems is CS independence, which is driven by the value function that portrays the individual objectives. The ontology provides the information necessary for game theory-based analysis. This is important in understanding SoS from the point of view of incentives (Axelsson, 2019). The extension of traditional game theory to hypergames is particularly relevant (Kovach et al., 2015). It captures that agents can have different perceptions of the situation in a similar way as in the proposed ontology.

Other kinds of dynamic analyses, such as simulations, are also possible from the information in the ontology.

A key part of SE is design space exploration. This can be seen as a sequence of model transformations, and the ontology gives guidance to what transformations are possible. The two key types of transformations are elaborations, where information is added to the model, and abstractions, where information is removed. Important elaborations are to add elements, properties, or relations; refine elements by turning them into systems with parts; and expand the value range of a property. Abstraction transformations are the inverse of these.

The basic system ontology and the agent extension clarify some differences between general SE and SoSE. In the development of integrated systems, the agents that create models (e.g., the engineers) are typically outside the system-of-interest which means that agent models are not essential. The engineers have strong incentives to align their models since the result is to be one integrated system. Therefore, model differences tend to be small, and the system can be taken as an objective reality with less need to separate models from what they represent. In SoSE, all these aspects are different. The CS are agents that are part of the SoS; they have individual purposes for their models with sometimes weak incentives to align; and separating models from reality is essential in understanding the SoS behavior. Building on the discussion of Jackson & Keys (1984), it is thus possible to see SE as a special case of the more general SoSE.

Addressing System-of-Systems Engineering Challenges

As mentioned in the introduction, a list of SoSE pain points was identified a decade ago (Dahmann, 2014). I will now revisit these to see how the proposed ontology can contribute to progress in the field:

- **SoS Authorities.** An SoS has multiple authorities, which makes decision-making and control much more complex. The ontology addresses this by making authorities explicit as agents. Therefore, their drivers and views of the world can be elicited and reasoned about to better understand the ensuing dynamics.
- **Leadership.** The development of an SoS is cross-organizational and hampered by differences in objectives, culture, and cognitive biases. The ontology makes it possible to articulate these differences in world models and reason about how to adapt leadership and communication practices to the particular SoS situation at hand.
- **CS perspective.** Many SoS include already existing systems, that are adapted to become CS. A key concern is how to design the SoS to simplify CS adaptation, thereby increasing the attractiveness of joining the SoS. The ontology makes it possible to describe the initial status of the system, and the transformations needed to make it into a CS, hence providing an improved understanding of SoS-level design decisions.
- **Capabilities and requirements.** Whereas integrated system development is driven, ideally, by clearly stated requirements, SoS design involves the combination of CS capabilities described on a higher level of abstraction. A capability can be seen as something a system can do, and in the agent ontology, this is a result of the actions, which are driven by the

world model, value function, and methods. This allows more details to be provided about what capabilities are available and hence enables a more exact analysis.

- **Autonomy, interdependence, and emergence.** It is difficult to predict SoS emergent properties due to the heterogeneity and interdependence of the CS. By providing a richer ontology that captures key characteristics related to CS independence through the concept of agents, better models can be created which leads to an improved understanding of the effects that emerge from SoS composition and dynamics.

- **Testing, validation, and learning.** SoS continue to evolve while in operation, and it is difficult to test and validate them beforehand. This is captured in the ontology by the fact that CS are agents that can evolve and learn through improvements in their world models, methods, and value functions.

- **SoS principles.** The final pain point, which is directly addressed by this paper, is the principles for SoS thinking. A solid ontology for describing SoS allows the representation of key characteristics of independent CS and is necessary for any progress in the field.

Underlying many of the pain points is the higher complexity attributed to SoS. Complexity is often seen as an inherent property of a system, but as I have shown in this paper, systems exist in the model domain. Since models are created by an agent for a certain purpose, complexity should be regarded as subjective, and not an objective fact (see also Jackson & Keys, 1984). By making the system-of-interest a part of a model, we can apply objective measurements of complexity to the system description, based on known metrics for computational and information complexity. The complexity is thus not in reality but chosen and in the eye of the beholder.

One of the reasons that SoS are often perceived as complex is the fact that SoS models need to contain CS agents with their internal models. To fully understand some aspects of the SoS, it is necessary to take the feedback loops among these alternative models into account. A single model, based on, e.g., known first principles of physics, does not suffice in socio-technical problems. In the terminology of Jackson & Keys (1984), SoS problems are in general systemic-pluralist and thus require soft systems methods.

Conclusions

In this essay, I have discussed the fundamental concepts needed to model systems-of-systems, master their complexity, and make informed design decisions. The factor that differentiates an SoS from an integrated system is the independence of its CS. This was captured in the proposed ontology as agents, who can choose actions to maximize their value.

A key insight is that agents themselves need to have models, so a model of an agent contains a model that may contain elements that represent other agents and their internal models. This nesting and intertwining of models may seem to result in an insurmountable complexity, but it is nevertheless a necessity to fully understand the resulting behavior of an SoS. There is no point in sweeping essential characteristics under the rug, but they must instead be embraced and techniques for handling them must be sought.

A basic ontology is a first step towards better SoSE practices, and it has been argued in the paper that this addresses many of the known SoSE pain points. The ambition of the ontology presented in the paper is to be a tool to explain the key ideas herein. To take the ontology into practical use, further refinements in its details are most likely required. Tools and techniques that manage the information effectively remain to be developed, and on top of these, analysis techniques are needed that can predict key characteristics as a basis for design decisions.

Acknowledgments

This research was funded by the Knowledge Foundation (KKS) under grant no. 2020-0230.

References

Ackoff, R. L. (1971). Towards a System of Systems Concepts. Management Science, 17(11), 661–671.

Apostel, L. (1960). Towards the formal study of models in the non-formal sciences. Synthese, 12(2), 125–161.

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

Axelsson, J. (2002). Towards an Improved Understanding of Humans as the Components that Implement Systems Engineering. INCOSE International Symposium, 1137–1142.

Axelsson, J. (2019). Game theory applications in systems-of-systems engineering: A literature review and synthesis. 17th Annual Conference on Systems Engineering Research (CSER), 154–165.

Axelsson, J. (2020). Achieving System-of-Systems Interoperability Levels Using Linked Data and Ontologies. INCOSE International Symposium, 651–665.

Axelsson, J. (2022). What Systems Engineers Should Know About Emergence. INCOSE International Symposium, 1070–1084.

Axelsson, J., & Eriksson, P. (2023). Higher-Level Capabilities of System-of-Systems Constituents: A Case of Industrial Ecosystems. 18th Annual System of Systems Engineering Conference.

Axelsson, J., & Svenson, P. (2022). On the Concepts of Capability and Constituent System Independence in Systems-of-Systems.17th Annual System of Systems Engineering Conference, 247–252.

Baxter, G., & Sommerville, I. (2011). Socio-technical systems: From design methods to systems engineering. Interacting with Computers, 23(1), 4–17.

Box, G. E. P. (1976). Science and Statistics. Journal of the American Statistical Association, 71(356), 791–799.

Checkland, P. (1993). Systems thinking, systems practice. Chichester, England: John Wiley & Sons.

Crutchfield, J. P. (1994). The calculi of emergence: Computation, dynamics and induction. Physica D: Nonlinear Phenomena, 75(1), 11–54.

Dahmann, J. (2014). System of Systems Pain Points. Proc. INCOSE International Symposium, 108–121.

Dori, D., et al. (2020). System Definition, System Worldviews, and Systemness Characteristics. IEEE Systems Journal, 14(2), 1538–1548.

ISO/IEC/IEEE. (2019). Standard 21841 Systems and software engineering—Taxonomies of systems-of-systems. ISO/IEC/IEEE.

Jackson, M. C., & Keys, P. (1984). Towards a System of Systems Methodologies. Journal of the Operational Research Society, 35(6), 473–486.

Kovach, N. S., Gibson, A. S., & Lamont, G. B. (2015). Hypergame Theory: A Model for Conflict, Misperception, and Deception. Game Theory.

Landauer, R. (1961). Irreversibility and Heat Generation in the Computing Process. IBM Journal of Research and Development, 5(3), 183–191.

Louie, A. H. (2010). Robert Rosen's anticipatory systems. Foresight, 12(3), 18–29.

Lu, J., Ma, J., Zheng, X., Wang, G., Li, H., & Kiritsis, D. (2022). Design Ontology Supporting Model-Based Systems Engineering Formalisms. IEEE Systems Journal, 16(4), 5465–5476.

Macal, C. M., & North, M. J. (2005). Tutorial on agent-based modeling and simulation. Proceedings of the Winter Simulation Conference, 2–15. IEEE.

Maier, M. W. (1996). Architecting Principles for Systems-of-Systems. INCOSE International Symposium, 565–573.

Minsky, M. L. (1965). Matter, Mind and Models. Proc. International Federation of Information Processing Congress, Vol. 1, 45–49.

Reynolds, M., & Holwell, S. (Eds.). (2010). Systems Approaches to Managing Change: A Practical Guide. London: Springer.

Rothenberg, J. (1989). The Nature of Modeling. In AI, Simulation and Modeling, 75–92. John Wiley & Sons, Inc.

Rousseau, D., Billingham, J., & Calvo-Amodio, J. (2018). Systemic Semantics: A Systems Approach to Building Ontologies and Concept Maps. Systems, 6(3).

Shannon, C. E. (1948). A mathematical theory of communication. The Bell System Technical Journal, 27(3), 379–423.

Sillitto, H., et al. (2019). Systems Engineering and System Definition. INCOSE.

Simon, H. A. (1955). A Behavioral Model of Rational Choice. The Quarterly Journal of Economics, 69(1), 99–118.

Stachowiak, H. (1973). Allgemeine Modelltheorie. Wien: Springer-Verlag.

Yang, L., Cormican, K., & Yu, M. (2019). Ontology-based systems engineering: A state-of-the-art review. Computers in Industry, 111, 148–171.

Biography

Jakob Axelsson received an MSc in computer science in 1993, followed by a Ph.D. in computer systems in 1997, both from Linköping University, Sweden. He was in the automotive industry with the Volvo Group and Volvo Cars from 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 include all aspects of systems-of-systems engineering. Prof. Axelsson is a member of INCOSE and has served as chairman of the Swedish chapter.