Towards an Ontology of Type-Level Phenomena for System Modeling

Rodrigo Fernandes Calhau[1,2,3], João Paulo A. Almeida[2], Giancarlo Guizzardi[3], Raquel Hoffmann[4], Luís Ferreira Pires[3], James Logan[5], and João Rebelo Moreira[3]

[1] LEDS Research Group, Federal Institute of Espírito Santo, Brazil
calhau@ifes.edu.br
[2] Ontology & Conceptual Modeling Research Group, Federal University of Espírito Santo, Brazil
jpalmeida@ieee.org
[3] Semantics, Cybersecurity & Services (SCS) Group, University of Twente, The Netherlands
{g.guizzardi, l.ferreirapires, j.luizrebelomoreira}@utwente.nl
[4] System Design and Management Graduate School, Keio University, Japan
raquel.hoffmann@keio.jp
[5] Ontogenesis Solutions, LLC
jlogan@ontogenesis-solutions.com

Abstract. In this paper, we propose a well-founded domain-independent system ontology based on the Unified Foundational Ontology (UFO) and the associated Multi-Level Theory (MLT). In our ontology, (composite) system types are designed by defining component types, connection types, and other type-level phenomena. We discuss how the proposed ontology can be used to guide the modeling of systems in specific domains. We position our work with respect to a number of ontologies in the literature that address system-related notions, as well as with respect to system modeling approaches.

Keywords: systems · ontologies · system modeling · conceptual modeling.

1 Introduction

Modern system engineering faces increasing challenges due to the inherent complexity of emerging technologies (e.g., AI, IoT, Advanced Robotics, Software Agents) and their combination. Among these challenges, there is the growing demand for integrating fundamentally different types of components forming a system. For example, combining hardware and software components forming cyber-physical systems, or combining computational, organizational and human parts forming cyber-social systems.

Modern system engineering requires better theories, tools, techniques, and methods for their development and maintenance. In particular, it requires proper support for the *modeling* of: complex systems, their composition (their parts), their structure (the interrelations between the parts), as well as capabilities and vulnerabilities that emerge from the interaction between these parts.

Modeling in this context can be challenging for many reasons. For example, understanding system decomposition (i.e., how a system is decomposed into parts) is linked to

complex phenomena such as (property and behavioral) emergence (e.g., a system with safe parts does not necessarily equal a safe system, since this depends on the kind of relationships and interactions among these parts).

Systems science researchers have studied these phenomena since General Systems Theory (GST) [1, 8, 12], identifying common principles applicable to different types of systems. However, these principles do not always result in practical guidelines that explicitly support complex system modeling.

Well-founded reference ontologies have been successfully employed to improve the expressivity, *domain appropriateness* and *comprehensibility appropriateness* of models and modeling languages in different domains (e.g. [4, 26, 32]). There is evidence that they can also play a fundamental role in providing explicit foundations for system modeling [39, 42]. In fact, many ontologies have been proposed over the years to model different types of systems [3, 5, 33, 44]. However, these tend to focus on issues that are specific to individual fields, and fail to address a broader system perspective.

In this paper, we propose a *well-founded multi-level system ontology*, and explore some of its *implications for ontology-based systems modeling*. Our ontology incorporates concepts from formal system ontology [12], GST [1, 8] and Systems Engineering [21]. We build on a *system core ontology* based on GST to explain the phenomenon of capability emergence [13, 14]. We extend that work to approach *type-level* phenomena in this domain. For example, (composite) system *types* are designed by defining component *types*, connection *types*, and other relevant concepts. These types are specified at varying levels of abstraction in the system engineering lifecycle, ranging from abstract functional specifications to detailed (physical) design.

We base our ontology on the Unified Foundational Ontology (UFO) [27]. UFO offers a set of domain-independent formal micro-theories that have been successfully applied in a variety of domains and tasks [25], in particular, in the revision of conceptual modeling constructs and languages. One of these micro-theories that has been incorporated into UFO is MLT (Multi-Level Theory) [16, 17], which is used here to support the proper modeling of important reified types (e.g., system, component, and connection types). To represent the proposed systems ontology, we employ the UFO-based conceptual modeling language OntoUML [26].

We have two ultimate application goals in mind for our ontology, namely, to use it to: (i) guide the definition of domain-specific system models by combining, specializing, instantiating, and refining the axioms of the ontology; (ii) establish a semantic basis for system modeling language revision, integration and reengineering.

The remainder of the paper is further structured as follows: Section 2 provides germane background on system theory; Section 3 introduces UFO and a preliminary system core ontology [14], both of which are employed in Section 4 to develop the multi-level system ontology; Section 5 elaborates on how this ontology can be used to support system modeling; Section 6 discusses some related work; Finally, Section 7 presents final considerations.

2 Systems and Their Components

Despite the significant variation in definitions [21], the majority of authors writing on the topic use the notion of system in a sense that reflects the original formulation of

General-System Theory (GST) [1,8,12], namely, taking the term to refer to a "complex" (or "structured") whole made up of "linked" (or interacting) parts. Accordingly, a system is viewed as a group of items that, via their relationships (or interactions), produce something novel, like behavior or emergent properties [8,21]. To support this understanding, Dori and Sillitto [21] provide the generic definition of a system as follows (based on a thorough literature review that examined over a hundred definitions): *"A system is a set (...) of parts (...) combined (...) in a way that creates (...) properties (...) not possessed (...) by the separated (...) parts (...)."* (we use ellipses ... here to simplify the presentation of the definition and omit the many alternative terms they have captured in their proposal.)

The generic definition provided by Dori and Sillitto [21] capture the main facets of the "system" concept. In addition to referring to the existence of properties that characterize the system *as a whole*, i.e., properties that go beyond the properties of parts in isolation, the definition also alludes to the inevitable *plurality* of things that in *combination* form a system ("a set of parts combined...").

Bunge [12] was one of the philosophers who dediated several writings to this notion. He used the term "global properties" to refer to system properties and divided them into: *emergent* and *resultant*. Properties that can be broken down, clarified, or reduced to the characteristics of a system's constituent parts are termed *resultant*. For instance, the simple sum of the mass of each constituent part of a system determines the system's total mass. Conversely, emergent properties are those that are not present in isolation in the separated parts, even though they are somehow grounded on the properties of parts. An example of emergent property is a ship's buoyancy. The buoyancy of the whole ship cannot be explained away by the buoyancy of the ship's individual components. In fact, most sections of a ship's steel hull are not buoyant by themselves. Buoyancy emerges from the combination of the various parts (including many non-buoyant ones).

Systems are created or developed to facilitate specific *functions* in their context. The notion of *function* is typically present in a teleological view of systems [37]. Accordingly, *functions* are demonstrated by a targeted outcome or behavior (action, process) [1,18]. The description of the *functions* of a system frequently begins by identifying *macro-functions* of the whole system, which are broken down into *sub-functions* (which are typically assigned to the components) [18,37], and so on recursively.

System structure, according to Bunge [12], is the collection of all connections among a system's components. He considers that the relationships between system elements can be either *"bonding"* or *"non-bonding"* [12]. Bonding relationships, also known as connections, are those in which one element influences and changes the other in some way [12]. The movement of information, material, or energy between elements can also be used to describe these relationships [12]. The *bonding structure* (or "configuration") [12] of a system is shaped by the set of *bonding* relationships that link all of its elements at a given time. The form, organization, or arrangement of the system is represented by the *bonding structure* in this context [21]. Bunge [12] argues that for an object to be regarded as a system, it must have a *bonding structure*. Similarly, Ackoff [1] considers that every component of every system must be directly or indirectly related to every other component. Therefore, there cannot be an isolated subset of elements in a system. Accordingly, elements that are wholly independent or unrelated should not be regarded as a system [37]. To put it another way, systems are

integral wholes in the sense of Simons [36], so they need a unifying condition that identifies and binds their components to each another.

3 Foundational Baseline

Our reference ontology of systems is designed as a specialization of domain-independent notions defined by the Unified Foundational Ontology (UFO). Here we focus on those notions that are germane to the purposes of this paper. For a full presentation of UFO, including axiomatization, we refer to [10, 26, 27]. UFO has two main taxonomies: one concerning types of *individuals* and another concerning types of *types*.

UFO categorizes individuals into *endurants*, *events*, and *situations*. *Endurants* are individuals that endure in time and may change qualitatively while keeping their identity. *Events* are individuals that 'occur' or 'happen' in time (e.g. 1970 Football World Cup Final in Mexico). *Situations* are particular configurations of a certain part of reality. There are two categories for *Endurants* in UFO: *objects* and *moments*. *Objects* are *endurants* that exist independently of another *individual* (e.g., Elvis Presley, the Earth, the Rolling Stones) . In contrast, *moments* (objectified or particularized properties) rely on another *individual* to exist (e.g., the yellowish color of the Sun, Martha's headache, Gerard's height). A particular kind of *moment* that can link two or more entities is a *relator* (e.g., Bonnie and Clyde's marriage, Paul McCartney and John Lennon's partnership). It reifies the moments that objects (the *relata*) have in the context of a relationship. An *intrinsic moment*, in contrast, depends existentially on a single individual. This includes both *modes* and *qualities*, the latter of which can be conceived of in a value space, such as color, height, weight, etc. *Modes* , in their turn, are intrinsic moments that are not directly projected into a value space. They encompass *dispositions*, which are moments that can manifest themselves through *events* in specific *situations*: examples include Pelé's ability to play football, which manifested in the 1970 World Cup; and Apollo 11's capability to travel in space manifested itself in the journey to the Moon in 1969.

UFO does not include the category *system*. It does, however, include the mereologically comparable notion of *functional complex*. Functional complexes are *objects* whose components play *functional roles* in its structure. In this case, whole and parts are bound by the *"component of"* relation [26]. A relation that is irreflexive, anti-symmetric and non-transitive [26], where non-transitivity means that transitivity may hold for certain cases but not for others. For example, the heart of the human being who plays the role of lead singer in a band, while a component of the person, is not a component of the band. In contrast, the piston of an engine that is a component of a car is also considered to be a component of that car. Moreover, like all other proper parthood relations, "component of" includes the *weak supplementation principle*, i.e., if x is a part of y then there must be a z mereologically disjoint of x, which is also part of y.

Differently from *individuals*, *types* are predicative entities which establish common properties for their instances. According to the ontological nature of their instances, the UFO taxonomy of types includes *endurant types* (e.g., "Person", "Marriage"), *event types* (e.g., "Wedding", "Football Match"), *object types* (e.g., "Organization", "Car"), *moment types* (e.g., "Age", "Enrollment"), etc. Additionally, UFO classifies types based on their modal metaproperties, dividing them into: *rigid*, *semi-rigid*, and *anti-rigid*. *Rigid* types, such as *kinds* and *subkinds* (e.g., "Person", "Car", "Pineapple"), are those that apply to

their instances invariably over the course of their entire existence. An endurant must instantiate exactly one kind since kinds give their instances a principle of identity. A kind, e.g., "Dog", passes on an identity principle to its subkinds, e.g., "Poodle", "Collie". In turn, abstract rigid types called *Categories* capture common attributes of endurants of different kinds (e.g., "Physical Object").

Types that classify their instances contingently (or "dynamically") are known as "anti-rigid" types. These anti-rigid types, whose contingent classification conditions are relational (e.g., "Supplier", "Husband", "Employer", and "Customer") [26], are of particular interest to us here, since components of a system play distinct roles on it based on their relationships. To generalize roles and role mixins, we use 'role (mixin)'. An example of a role mixin is "Customer" that subsumes "Personal Customer" and "Organizational Customer". Role mixins are relational, anti-rigid types that represent common relational properties of objects that instantiate different kinds.

Types encompass *relations* among two or more *entities*. In UFO, a special type of relationship termed *material relation* requires the explicit representation of a link binding it to a corresponding relator type from that relation is derived. For example, the "married with" is derived from the relator type "Marriage"; the "enrolled in" relation from *enrollment*, and so on. At the instance level, a relationship exists between two or more *relata* (e.g. John is "married with" Mary) if there is a particular relator of a suitable type that binds them (in this case, a particular marriage). In material relations, the types of the associated relata are typically *roles* (in this case, the role "spouse").

In [14], some of us proposed a preliminary system core ontology based on UFO that discusses the primary distinctions among systems and their constituent parts from an individual standpoint. GST and Bunge's system ontology (in particular, his notions of composition, environment, structure, and mechanism model (CESM)) were used to identify the requirements for that ontology [12]. In order to distinguish *systems* from "simple" *objects*, that ontology takes into account the following factors: (i) the degree of complexity (number of components); (ii) the integrated structure; (iii) the complementarity and heterogeneity of the functional roles (and capabilities) of components; and (iv) the emergence of new properties and behaviors.

Figure 1 presents a fragment of the system ontology in OntoUML [26]. "System" is considered a *«category»* of *objects*, characterized by *system moments*, connected with external entities and with at least two proper parts (*components*). Note that the entities in the systems ontology specialize notions of UFO in its taxonomy of individuals, namely, the notions of "Object", "Moment" and "Situation".

Components are those objects that are part of a system. This includes entities usually referred to as *units*, *modules*, *interfaces*, etc. When a system is part of another system, it is termed a *subsystem* as depicted in Figure 1. This enables the recursive (hierarchical) decomposition of systems. The relation "has proper part" is of type *«componentOf»* (and hence, non-transitive, irreflexive, asymmetric, and obeying weak supplementation). It is also a case of non-exclusive parthood (hence the UML aggregation symbol), i.e., components can be part of more than one system.

Structure is captured via *bonding* and *non-bonding relations*, following Bunge [12]. Bonding relations are *«material»* relations. They can be used to: connect a system or a system component with *external entities*, i.e., entities in the environment ("is externally connected with"); connect components within the same system ("is internally connected with"). Figure 1 only shows the reification of internal connection relators, but the

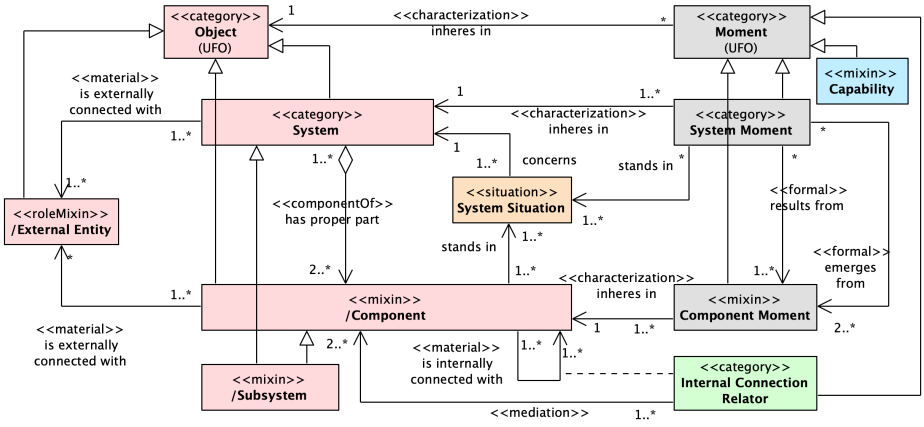

Fig. 1. Well-founded System Ontology proposed in [14]

same applies to external connections. Non-bonding relations, in turn, concerns system constraints and are represented using comparative relations from UFO (not illustrated in the figure).

The ontology addresses global properties [11] through *system moments*, as shown in Figure 1. It adopts the distinction between *emergent* and *resultant properties* discussed in [11]. In this view, emergence is not a simple phenomenon that appears from one and only factor. For example, (emergent) *system moments* are said to stand in specific *system situations* ("stands in" relation). This is meant to represent the complexity of the settings of emergence, mainly: system constraints, connections, component aspects (i.e., capabilities). Additionally, *system moments* "emerge from" (component) *moments* (including emerging *dispositions*, *qualities*, *relators*). The notion of *function* is reflected in the ontology via the notion of *capability*. We see *functions* as specialized *capabilities* (*dispositions*) that bring benefits to other entities in the system context [21, 30]. Both system and component moments may be capabilities too, i.e., one may talk about system capabilities and component capabilities.

4 Types of Systems, Components, and Connections

The ontology presented in the previous section captures invariant features of systems, their parts (components), and their properties (system and component moments). However, it is an ontology of individuals and, as such, it does not capture an important type-level dimension of this domain. If we understand the system design task as one of producing a system type, along with component types, connector types, etc., it becomes clear that the universe of inquiry of system design encompasses not only individuals but also types as first-class citizens.

System *types* are designed by establishing conditions for conforming systems, including their characterizing properties, the *types* of components they require, the *roles* components play in a conforming system, the *types* of connections they establish in the

scope of the system, etc. For example, the "Porsche 356 Carrera GT Speedster" (a system type) features a "1.6-Liter Four-Cam Engine", which plays the role of "Main Engine" in cars of that type. Instances of this type can be connected with external entities (e.g., with a "Driver" or with a "Gas Pump Nozzle") through certain types of connections.

Figure 2 details the taxonomy of the system type ontology, focusing on the subtypes of *object type* we are concerned with: *system type*, *component type*, and their specializations. An important feature of the ontology is the use of high-order types, i.e., types whose instances are themselves types. For example, instances of "System Type" are types that specialize the base type "System" (e.g., "Car"). This modeling strategy is based on Odell's powertype pattern as incorporated in the OntoUML profile, following its axiomatization in the MLT multi-level theory [15]. The pattern is formed by a high-order type stereotyped *«type»* related with a base type through an association with specialized semantics stereotyped *«instantiation»*. In the model, instances of the high-order type specialize the base type through a generalization set that has that high-order type as powertype (which in UML is represented with a colon ':' followed by that high-order type's name next to the generalization set). So, when designing a particular system type, e.g., "Car", one would specialize "System", indicating "System Type" as a powertype of the specialization's generalization set. This makes "Car" an instance of "System Type" (along with any other subclasses in the same generalization set). By following this pattern, a number of rules can be enforced on the model [15].

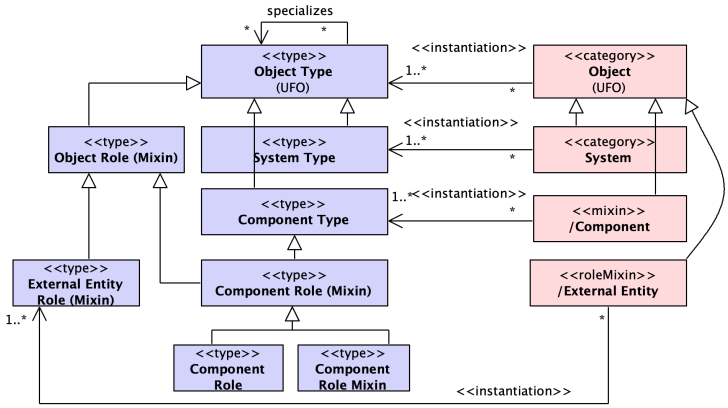

Fig. 2. System Type Ontology - Taxonomy

When defining a component type, such as a component role, one may establish the supertypes of that type in order to determine the conditions for componenthood. For example, a "Main Engine" (a "Component Role" in the scope of a "Car") is necessarily an "Engine" and hence, "Main Engine" specializes "Engine". More complex patterns can also be used to establish requirements for component role playing. For example, the role of "Blood Pump Component" in the human body's circulatory system may be played by a "Heart" or by an "ECMO Machine". In this case, we can establish a super type of "Heart" and "ECMO Machine" focusing specifically on the required function of

that component (i.e., “Blood Pump”). The “Component Type” is the locus of any other requirements that are imposed to objects as components of a system.

Figure 3 shows a fragment of the system type ontology, focusing on system composition at the type level. A “System Type” “is formed by” at least one “Component Type” and “is externally bonded to” “External Entity Roles (Mixins)”. A “Component Type” “is internally bonded to” at least one “Component Type”, which may be “externally bonded to” zero or more “External Entity Roles (Mixins)”.

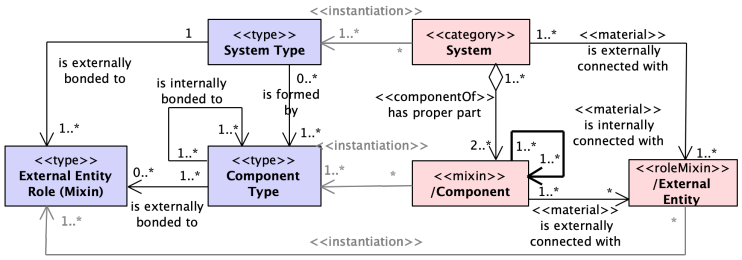

Fig. 3. System Type Ontology - System Composition and Structure

The association “is formed by” is an example of association with *deep characterization*: it affects not only the classification level below, in that instances of “System Type” must be related to at least an instance of “Component Type”, but it also affects the instances of subtypes of “System” and of “Component”. This is because “is formed by” is a *regularity property* [24]: it regulates “has proper part” by constraining the types that may be related through this association. Note that UML class diagrams do not have the expressive means to declare this kind of regulation in general. Hence, an additional constraint would be needed here. A syntactic constraint for models specializing the system type ontology is recommended: only types that are instances of “System Type” and “Component Type” (and hence specializations of “System” and “Component”) may have associations that specialize “has proper part” in this ontology. The association “has proper part” is considered *abstract* in an analogous way to abstract classes; it can only be used to link instances of “System” and “Component” through associations that specialize it (e.g., “has main engine”, “has chassis”). Specializations of “has proper part” determine the compatible system and component types (e.g., “Car” and “Engine”, “Car” and “Chassis”).

The same rules apply to “is internally bonded to”, as it regulates “is internally connected with” (and analogoulsy “is externally bonded to”). The semantics of “is internally bonded to” and of “is internally connected with” require that only component roles that form the same system type may be internally bound to each other; likewise, a component can only be internally connected with other components when their roles are internally bonded to each other.

Figure 4 shows a fragment of the system type ontology focusing on *Connection Relators*, from which connections are derived. As illustrated, “Relator” (UFO) instantiates “Relator Type” (UFO). At the type level, “Relator Type” has mediation with “Object Role (Mixin)”, which includes “Component Role (Mixin)” and “External Entity Role

(Mixin)", as previously presented. At the individual level, Relator (UFO) includes both "External Connection Relator", which mediates the relationship between components and external entities, and "Internal Connection Relator", which mediates the relationship between components.

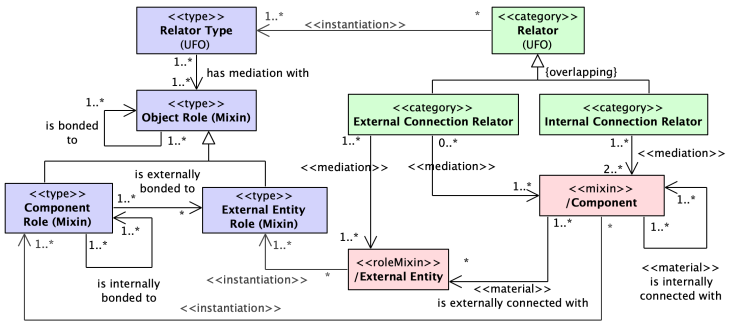

Fig. 4. System Type Ontology - Connection Type

5 Modeling Systems and their Components

In order to show some consequences of the ontological analysis and the ontology proposed in the previous section, we present in this section an application related to system modeling. Such a case is specifically related to system components and the detailing of connections. We propose some (integrated) viewpoints for ontology-based conceptual modeling. Then, we present an example of modeling using these views in OntoUML. We propose a number of viewpoints in alignment with Bunge's CESM (Component, Environment, Structure, and Mechanism) model [11]. According to this model, a system has a *composition* (formed by the set of components of the system), an *environment* (formed by the external entities that interact with the system), a *structure* (formed by the relationships/connections between its parts), and a *mechanism* (formed by the events/processes which happened in the system). Besides the viewpoints from CESM, we also included the *property* and *emergence* viewpoints of the system.

A ***system environment viewpoint*** reveals the types of external connections between the system and *external entities*. It identifies *system roles* and *external entity roles* that are played by the system and external entities, respectively, when they are connected. Supertypes of system role and external entity roles reveal the types of objects involved. This viewpoint may involve the reification of external connection relators. As mentioned, the ***system composition viewpoint*** focuses on the internal perspective at the component level, detailing the types of system components that form a system type. The viewpoint identifies component types (often roles) in the context of the system and specifies the types of objects that may be components of instances of the system type. A ***system structure viewpoint*** focuses on the connections between the components of a system. The main focus is on the internal perspective, however, the external perspective

can also be considered. All component types must be connected directly or indirectly. Internal connections can only occur between component types of the same system type. External connections only occur with entities external to the system. A ***system property viewpoint*** reveals system and component moments such as qualities and dispositions (e.g., capabilities and vulnerabilities). A ***system mechanism viewpoint*** reveals system and component events as a result of the manifestation of their properties (dispositions). Finally, a ***system emergence viewpoint*** reveals how system moment types and/or system connection relator types arise from or are derived from component moment types and/or component connection relator types.

To illustrate the application of the system ontology and proposed modeling viewpoints, we consider here a motorcycle as an example of a cyber-physical system, i.e., a physical system that can include cybernetic components. This is a recurrent example in the model-based systems engineering literature [9,23,43,45]. To illustrate the proposal, we will consider the motorcycle-biker system as part of an organization that needs to track and control a fleet of motorcycles. To do this, the organization needs a *fleet tracker* as its component. So, as a result, the motorcycle must have capabilities that allow it to be tracked by the fleet tracker.

This scenario can be represented by the *system environment viewpoint* as depicted in Figure 5. In this case, the environment viewpoint can be understood from two different perspectives: (a) considering the motorcycle and the fleet tracker as parts of an organization system;(b) from the perspective of the motorcycle as a system, without considering the organization. The distinctions from system type ontology help clarifying these different perspective. In the first one, illustrated in Figure 5 (a), 'Organization' specializes 'System' and both 'Motorcycle' and 'Fleet Tracker' specialize 'Component' as parts of that organization. In this case, the connection between an instance of 'Motorcycle' and the 'Fleet Tracker' is internal, with 'Tracking Connection Relator' specializing 'Internal Connection Relator' (not depicted). In the second perspective, in Figure 5 (b), 'Motorcycle' specializes 'System' and 'Fleet Tracker' specializes 'External Entity'. In this case, the connection between a motorcycle and the fleet tracker is external, with 'Tracking Connection Relator' specializing 'External Connection Relator'. In short, a motorcycle is a subsystem because it is both a system and a component. 'Fleet Tracker' is both an 'External Entity' and a 'Component' depending on the perspective.

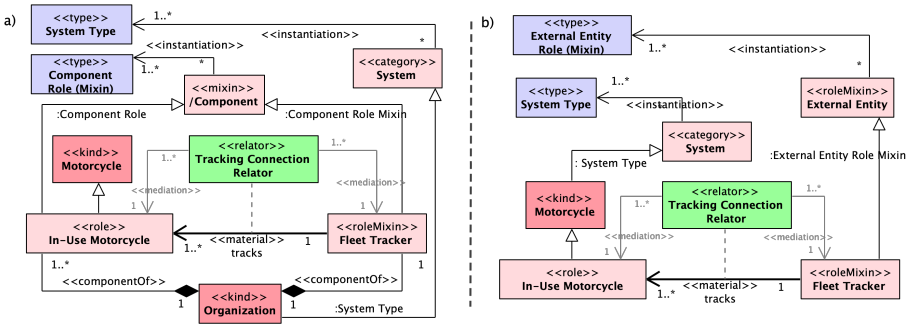

Fig. 5. System environment view (in OntoUML)

As shown, in both cases, the connections are mediated by a relator. In particular, the connection between the 'Fleet Tracker' and the 'In-Use Motorcycle' is mediated by the 'Tracking Connection Relator', from which the 'tracks' relation is derived. Reification allows us to treat the (external/internal) connection relator as an endurant, ascribing to it temporal properties (it is created/destroyed at certain points in time) and qualities (representing its characteristics). Moreover, these relators can serve as the basis for further decomposition (when parts of the connection relator are revealed) and can enable the detailing of emergent phenomena (when component connection relators are considered).

To allow the motorcycle to be tracked by the Fleet Tracker, the motorcycle must have appropriate components that enable this function. Figure 6 identifies the motorcycle component roles, forming the 'Motorcycle' system type (a kind). The composition of the motorcycle is divided into two levels: motorcycle subsystems (braking, tracking, acceleration, steering, etc.) and components, which are parts of these subsystems. The tracking subsystem is formed by 'Microcontroller Component' and 'GPS Tracker Component'. As depicted, there is an explicit separation between component types (in this case component roles) and their allowed types (Handlebar, Wheel, Frame, etc.). This allows us to identify component roles that may be played by objects of the same type (the case of 'Wheel' and 'Motorcycle Rear Wheel Component' and 'Motorcycle Front Wheel Component'.) A particular 'Wheel' is only contingently a motorcycle component. This separation of object type and component type allows us to stipulate requirements for objects as components in the scope of a certain system and in a certain role. For example, different tire pressures may be required for wheels functioning as 'Motorcycle Rear Wheel Component' or 'Motorcycle Front Wheel Component'. This viewpoint is also the locus for the specification of certain constraints, e.g., it is not the case that the same wheel may play both roles simultaneously in a motorcycle. Nevertheless, since roles classify objects dynamically, the possibility of replacement of components exists, unless constraints are specified. For example, one could rule out the replacement of the 'Motorcycle Frame Component' by deeming it an 'essential part' [26]. Note that we have omitted the powertypes (that were revealed in Figure 5) for the sake of readability.

When certain component roles can be played by objects of different kinds, the OntoUML role mixin pattern can be employed [26]. This can be illustrated in the motorcycle example. The motorcycle can have up to three distinct storage compartments: side storage compartment, rear storage compartment, and seat storage compartment. These subtypes of the storage compartment role mixin can be played, respectively, by instances of the rigid types: side bag, case (box), and seat. Note that a seat can play two roles: that of a Motorcycle Seat Component and of a Motorcycle Storage Component (more specifically Motorcycle Seat Storage Component).

The separation of object type and component type allows us to identify object types whose specializations form distinct system types. For example, an engine may play the role of 'Motorcycle Engine Component' or 'Tricycle Engine Component'. A constraint in this model is required to state that the same engine cannot play both roles simultaneously. The tricycle shares the same *type of engine* as the motorcycle. Given the dynamic aspect of roles, it is even possible for an engine to be swapped from a motorcycle to a tricycle (and vice versa).

Still considering the tracking issue, to allow the motorcycle to be tracked, it must have appropriate internal connections that enable such capability. Figure 7 shows a

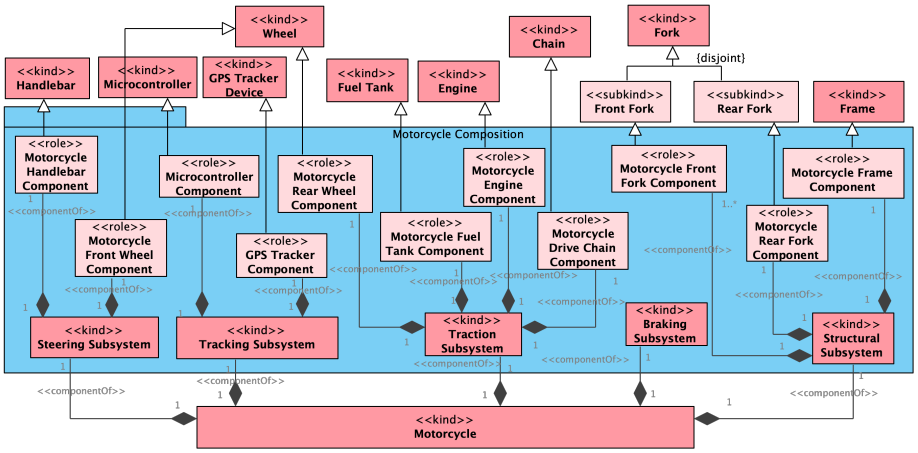

Fig. 6. System component view

system structure view. Each of the motorcycle components is connected with at least one different component. Furthermore, the connection with the Fleet Tracker is now refined, detailing how they happen at the component level. Two connections are identified: the Fleet Tracker "controls remotely" the motorcycle tracking subsystem (to block the motorcycle's engine, if necessary), and the 'GPS Tracker Component' "informs the location" of the motorcycle to the Fleet Tracker controls it. Although the example focuses on connections between components, the subsystems presented above are also connected (through the connections between their components).

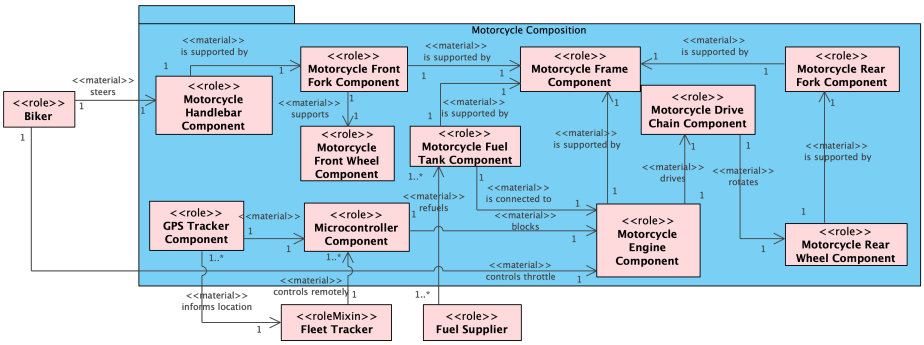

Fig. 7. System structure view

To understand the emergence of the tracking capability, we leverage the reification of connections in the emergence view, in order to show how system-level connections emerge. We first represent *connection relators*, as shown in Figure 8, for each of the connections with the Fleet Tracker. As shown in the figure, the 'Tracking Connection

Relator' emerges from two connection relators from the component level: *'GPS Tracking Connection Relator'* and *'Remote Controlling Connection Relator'*.

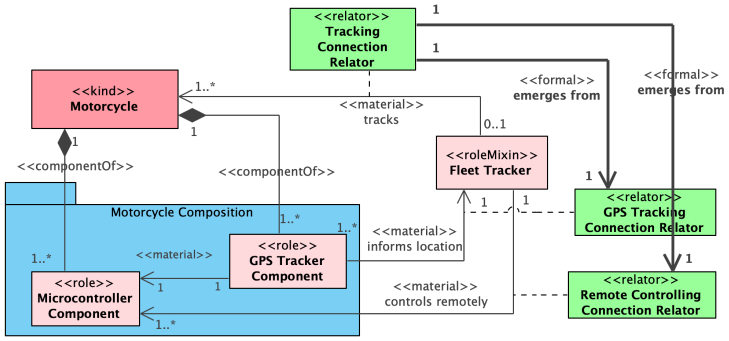

Fig. 8. System emergence view (connections)

Along with the emergence of connections presented above, there is also the emergence of capabilities. This is illustrated in Figure 9. As depicted, the motorcycle has the susceptibility to be tracked (a capability). This capability emerges from the capabilities of its parts, such as 'Motorcycle Controlling Capability' and 'Location Sending Capability'. On the other hand, the Fleet Tracker has the following capabilities: 'Remote Motorcycle Locating Capability' and 'Remote Motorcycle Controlling Capability'. As a result of the motorcycle's capability to be tracked and the capabilities of the Fleet Tracker, the 'Tracking Capability' of the whole system (the organization) emerges.

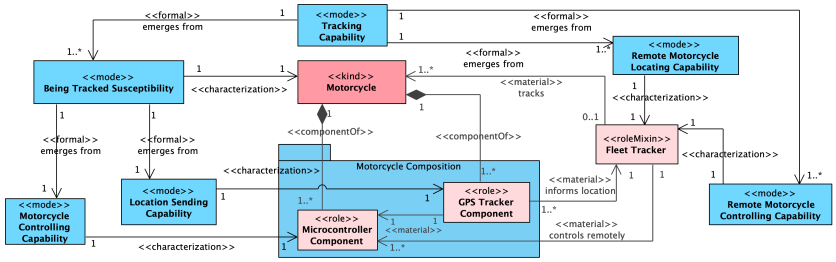

Fig. 9. System emergence view (capabilities)

6 Related Work

A number of ontologies have addressed systems and their components and connections. We review the most important of the proposals in the sequel, positioning them with

respect to their account of the various notions we have discussed in this paper (composition, functional decomposition, structure), especially in the type perspective. We focus on system ontologies that approach systems in a general sense, based on system science.

Based on Bunge's work, Wand et al. [40] proposed the Bunge-Wand-Weber (BWW) ontology. BWW has been an influential conceptual framework in the IS field, serving as a basis for a number of conceptual modeling efforts in the area [29]. BWW considers important distinctions (system, subsystem, etc) and also composition and structure. However, it does not address some aspects discussed here. Moreover, there is no way of capturing the *system type* perspective (including system types, component types, and component roles) in that framework. Doing so requires treating types and type relations as first-class citizens (higher-order types and higher-order relations), something which is not available to the BWW framework.

Lukyanenko et al. [29] present a new ontology called Bunge's Systemist Ontology (BSO). BSO is one of the most related works, since is also based on Bunge's work. BSO focuses on the concept of a *system*, which is the elementary unit of the ontology [29]. Compared with BWW, BSO further details the emergence phenomenon. BSO extends Bunge's CESM model to support conceptual modeling [29], adding emergent properties. This is similar to this work, which also added a viewpoint regarding emergence. One of the differences between this work and BSO is that here we consider multi-level distinctions such as system type and component type.

OntoCAPE is an ontology developed in the context of an open initiative that technically supports, through the use of standards, the problem of interoperability in engineering. OntoCAPE defines in its ontology some established concepts which are strongly based on GST concepts, such as: *system*, *subsystem*, *super-system*, *element*, *property*, *connection*, *requirement*, *function*, and *behavior* [31]. However, it lacks support for representing a *system-type* perspective, the phenomenon of emergence, component roles and system roles.

In [21] and [20], the authors propose what they call an "ontological framework", in which systems are modeled using the Object-Process Methodology (OPM) [19]. The OPM notation is based on an ontology that allows the representation of several important aspects related to systems such as whole, part, system, subsystem, and etc. Their approach lacks the support for modeling: the phenomenon of emergence, the system-type perspective, as well as a notion of component role.

Another relevant ontology in this context is the Industrial Ontology Foundry (IOF) [38]. IOF, like our ontology, is based on a foundational ontology, namely, BFO [34]. By leveraging BFO's support, it provides well-founded definition for concepts such as system, engineered system, mechanical system, function, capability, component, and component role. IOF is one of a few works considering capabilities and roles. However, it lacks support for the representation of emergence, as well as for explicitly representing connections as relators, as there is no equivalent notion to relator in BFO. The type perspective is not considered as well.

From the standpoint of system structure, the majority of the works take into account a type of structural relationship between systems or components through the following definitions: *relation* [22], *structural relation* [33], *connector* [19], *connection* [28, 31], *binding mutual* [40], *link* [5], or *interaction* [2, 7]. Concepts about mediators of some sort are also taken into account by other ontologies as interfaces, ports, or connection points [6, 28, 35, 41]. Most ontologies do not explicitly consider external connections

(i.e., between a component and an external entity). Generally, this kind of relationship is represented only between systems and not between components of a system.

7 Final Remarks

In this paper, we have present a well-founded multi-level systems ontology covering various aspects of systems, such as their environment, composition, functions, component roles, among other elements. The ontology is grounded on UFO [26] and was proposed based on GST principles, allowing the broad representation of systems including their composition, function, structure, and properties.

A key aspect in UFO (including MLT) is the modeling of types, doing justice to the distinction among different types of types (e.g., kinds, roles, role mixins), as well as supporting higher-order types. With the support of these theories, we could account not just for systems and components, but also for system types and component types, as well as their relations.

Important domain information often remains tacit in process standards, data item descriptions, and representation approaches used in system engineering. Transitioning from tacit to explicit information in this area is challenging, but key to advancing the theory and practice in the field. The ontological analysis and the resulting artifacts proposed here contribute to addressing this problem by making system concepts and constraints explicit. As a modeling contribution, we propose *modeling views and patterns* extracted from this ontology, and which can be reused and extended for support modeling in application domains.

In the future, we intend to extend this work to also consider behavioral aspects addressing the *Mechanism Viewpoint*. We also intend to provide clear guidelines for understanding and modeling system (de)composition in existing modeling notations (e.g., UML, SysML), with special attention to the emergence of desired capabilities.

A complete formalization of the proposed ontology in accordance with the axiomatization of UFO ([27]) is planned as future work. This formalization will also enable us to support automated verification, validation and reasoning over system models extending this ontology. In system modeling, a number of constraints must be attended in order to guarantee the consistency of complex system specifications (e.g., if a system is connected to an external entity, at least one of its components must also be connected to it; certain component connections demand components of certain types). This is especially important for creating constraints for some classifications of systems, due to their nature. For example, cyber-physical systems must be formed only by cybernetic and physical components, and must not be formed by social components; socio-technical systems must necessarily have human components (people); physical systems cannot emerge from social components; or, components of physical systems must not have social connections but only physical connections. There are other more general formalizations that can be useful in modeling all types of systems, generating important constraints. For example, a system cannot emerge from itself or from the system of which it is a part; the emergence relationship cannot be cyclic; and so on.

Acknowledgment

This study was supported in part by CNPq (443130/2023-0, 313412/2023-5), FAPES (1022/2022) and the DSYNE INTPART network (Research Council of Norway project number 309404).

References

1. Ackoff, R.L.: Towards a system of systems concepts. Management Science **17**(11), 661–671 (Jul 1971)
2. Akkiyat, I., Souissi, N.: Process meta model extended for the improvement cycle in a SoS context. In: 2019 4th World Conference on Complex Systems (WCCS). IEEE (Apr 2019)
3. Aquino, N.M.R.: A smart assessment of business processes for enterprises decision support. Université de Lorraine (2021), ph.D. thesis
4. Azevedo, C., Almeida, J.P.A., van Sinderen, M., Ferreira Pires, L.: Towards Capturing Strategic Planning in EA. In: 2015 IEEE 19th International Enterprise Distributed Object Computing Conference. pp. 159–133. IEEE (2015). https://doi.org/10.1109/EDOC.2015.31
5. Baek, Y.M., et al.: A meta-model for representing system-of-systems ontologies. In: 6th IEEE/ACM International Workshop on Software Engineering for Systems-of-Systems, SESoS@ICSE 2018, Gothenburg, Sweden, May 29, 2018. pp. 1–7. ACM (2018)
6. Bakirtzis, G., et al.: An ontological metamodel for cyber-physical system safety, security, and resilience coengineering. Software and Systems Modeling **21**(1), 113–137 (Jun 2021)
7. Banzhaf, W., et al.: Defining and simulating open-ended novelty: requirements, guidelines, and challenges. Theory in Biosciences **135**(3), 131–161 (May 2016)
8. von Bertalanffy, L.: General system theory. George Braziller, New York (1968)
9. Biggs, G., et al.: Integrating safety and reliability analysis into mbse: overview of the new proposed omg standard **28**(1), 1322–1336 (2018)
10. Botti Benevides, A., Bourguet, J.R., Guizzardi, G., Peñaloza, R., Almeida, J.P.A.: Representing a reference foundational ontology of events in sroiq. Applied Ontology **14**(3), 293–334 (2019)
11. Bunge, M.: Treatise on Basic Philosophy: The Furniture of the World. Ontology I. Reidel Pub. (1977)
12. Bunge, M.: Treatise on Basic Philosophy. Ontology II: A World of Systems. Springer Netherlands, Dordrecht, Netherlands (1979)
13. Calhau, R.F., et al.: Exploring system behavior in a system ontology. In: 2023 ER Forum, Demo and Posters, ER Forum 2023, co-located with 42th International Conference on Conceptual Modeling, ER 2023 (2023)
14. Calhau, R.F., et al.: A system core ontology for capability emergence modeling. In: Enterprise Design, Operations, and Computing. p. 3–20. Springer Nature Switzerland (Oct 2023)
15. Carvalho, V., et al.: Using a Well-Founded Multi-Level Theory to Support the Analysis and Representation of the Powertype Pattern in Conceptual Modeling. In: Proc. CAiSE 2016. LNCS, vol. 9694, pp. 309–324. Springer (2016)
16. Carvalho, V., et al.: Multi-level ontology-based conceptual modeling. Data & Knowledge Engineering **109**, 3–24 (2017)
17. Carvalho, V., et al.: Toward a well-founded theory for multi-level conceptual modeling. Software and Systems Modeling **17**, 205–231 (2018)
18. Compagno, F., et al.: Towards a formal ontology of engineering functions, behaviours, and capabilities. Semantic Web Journal (forthcoming)
19. Dori, D.: Object-process methodology. In: Encyclopedia of Knowledge Management, Second Edition, pp. 1208–1220. IGI Global (2011)

20. Dori, D.: Object-process methodology for structure-behavior co-design. In: Handbook of Conceptual Modeling, pp. 209–258. Springer Berlin Heidelberg (2011)
21. Dori, D., et al.: What is a system? an ontological framework. Syst. Eng. **20**(3), 207–219 (2017)
22. Duan, H., et al.: The development of a digital twin concept system. Digital Twin **2**, 10 (Aug 2022)
23. Eckl, C., et al.: Using the "model-based systems engineering" technique for multidisciplinary system development. In: ISPE CE. pp. 100–109 (2015)
24. Guizzardi, G., et al.: Towards an Ontological Analysis of Powertypes. In: Proceedings of the Joint Ontology Workshops 2015 Episode 1: The Argentine Winter of Ontology. CEUR Workshop Proceedings, vol. 1517. CEUR-WS.org (2015)
25. Guizzardi, G., et al.: Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story. Applied Ontology (Online) **10**, 259–271 (2015)
26. Guizzardi, G.: Ontological Foundations for Structural Conceptual Models. No. 15 in Telematica Instituut Fundamental Research Series, Telematica Instituut, The Netherlands (2005)
27. Guizzardi, G., et al.: UFO: Unified foundational ontology. Applied Ontology **17**(1), 167–210 (Mar 2022)
28. Lefrançois, M.: Planned ETSI SAREF Extensions based on the W3C&OGC SOSA/SSN-compatible SEAS Ontology Paaerns. In: Workshop on Semantic Interoperability and Standardization in the IoT, SIS-IoT. p. 11p. Amsterdam, Netherlands (Sep 2017)
29. Lukyanenko, R., Storey, V.C., Pastor, O.: System: A core conceptual modeling construct for capturing complexity. Data & Knowledge Engineering **141**, 102062 (2022)
30. Merrell, E., et al.: Capabilities (2022)
31. Morbach, J., et al.: Onto CAPE 2.0 —a (re-)usable ontology for computer-aided process engineering. In: Computer Aided Chemical Engineering, pp. 991–996. Elsevier (2008)
32. Nardi, J.C., de Almeida Falbo, R., Almeida, J.P.A., Guizzardi, G., Pires, L.F., van Sinderen, M.J., Guarino, N., Fonseca, C.M.: A commitment-based reference ontology for services. Information Systems **54**, 263–288 (Dec 2015). https://doi.org/10.1016/j.is.2015.01.012
33. Naudet, Y., et al.: Towards a systemic formalisation of interoperability. Computers in Industry **61**(2), 176–185 (feb 2010)
34. Otte, J.N., et al.: BFO: Basic formal ontology1. Applied Ontology **17**(1), 17–43 (Mar 2022)
35. Pfister, F., et al.: A design pattern meta model for systems engineering. IFAC Proceedings Volumes **44**(1), 11967–11972 (Jan 2011)
36. Simons, P.: Parts: A Study in Ontology. Oxford University Press (08 2000)
37. Skyttner, L.: General systems theory: ideas & applications. World Scientific (2001)
38. Smith, B., et al.: A first-order logic formalization of the industrial ontologies foundry signature using basic formal ontology. In: Proc. Joint Ontology Workshops (JOWO) 2019 Episode V:. CEUR Workshop Proceedings, vol. 2518. CEUR-WS.org (2019)
39. Van Ruijven, L.: Ontology and model-based systems engineering. Procedia Computer Science **8**, 194–200 (2012)
40. Wand, Y., et al.: On the deep structure of information systems. Information Systems Journal **5**(3), 203–223 (Jul 1995)
41. Yang, D., et al.: Development of a product configuration system with an ontology-based approach. Computer-Aided Design **40**(8), 863–878 (Aug 2008)
42. Yang, L., Cormican, K., Yu, M.: Ontology-based systems engineering: A state-of-the-art review. Computers in Industry **111**, 148–171 (2019)
43. Yildirim, U., Campean, F.: Functional modelling of complex multi-disciplinary systems using the enhanced sequence diagram. Research in Engineering Design **31**(4), 429–448 (2020)
44. Yilma, B.A., et al.: Systemic formalisation of cyber-physical-social system (CPSS): A systematic literature review. Comput. Ind. **129**, 103458 (2021)
45. Zheng, P.: One-of-a-Kind Product Development for Mass Personalisation in a Cloud-Based Environment. Ph.D. thesis, ResearchSpace@ Auckland (2017)