




# Ontological Analysis of Advanced Capability Modeling in ArchiMate: A First Step Towards Language Revision

Rodrigo F. Calhau<sup>1,2,3</sup> , João Paulo A. Almeida<sup>1</sup> , and Giancarlo Guizzardi<sup>3</sup> 

<sup>1</sup> Ontology & Conceptual Modeling Research Group, Fed. Univ. of Esp rito Santo, Brazil

<sup>2</sup> LEDS, Federal Institute of Esp rito Santo, Serra, Brazil

<sup>3</sup> Semantics, Cybersecurity & Services, University of Twente, Enschede, The Netherlands  
calhau@ifes.edu.br; jpalmeida@ieee.org; g.guizzardi@utwente.nl

**Abstract.** In order to support capability management, the field of Enterprise Architecture proposes methods and notations to model enterprises and their capabilities. ArchiMate is one of such notations and includes constructs to support capability mapping and other capability management tasks. However, the notation lacks some fine-grained distinctions that are required to understand intricate phenomena involving capabilities, including capability *interaction* and the *emergence* of capabilities. In this work, we perform an ontological analysis of the language’s support for capability modeling based on a well-founded ontology of capabilities aligned with the Unified Foundational Ontology (UFO). Through this ontological analysis, we identify some issues outlining possible improvements for ArchiMate. This is a first step towards language redesign, which may include the proposal of language patterns and/or the revision of language constructs.

**Keywords:** ArchiMate · ontological analysis · capabilities · emergence.

## 1 Introduction

Enterprises and organizations have been facing challenges with the rapid and unpredictable development of new technologies that impact their operating environments. With these fast changes comes the need for the development of organization-wide capabilities, which emerge from a well-balanced combination of organizational elements.

Given the importance of organizational capabilities, it is no surprise that they have been given attention in disciplines such as capability management and Enterprise Architecture (EA) [6]. In particular, EA aids the capability management process by offering structured visualizations that align strategic goals with IT and business processes, identifying gaps and opportunities for improvement [18]. Notations such as ArchiMate [41] have been widely adopted to support practices such as business and information technology capability planning and mapping. With this notation, it is possible to construct models to support capability management and visualize important aspects such as capability decomposition and other types of capability relationships [20, 42].

Capability modeling faces a series of demanding challenges, especially with regard to how to decompose and compose capabilities, how to trace, relate (or combine), and compare them. Although currently available notations offer resources to represent some aspects of capabilities such as relationships of composition, association, impact,

dependence, and realization, they do not fully address the phenomenon of capability emergence. In other words, existing notations may be effective in representing isolated capabilities, but they do not provide a robust framework for dealing with the complexity inherent in modeling capabilities in a broader context.

Some of the limitations of existing notations can be traced to their underlying ‘world view’. Making this world view explicit and accounted for is the objective of *ontological analysis*, which has been proposed as a means to evaluate the expressiveness and domain appropriateness of conceptual modeling languages [44]. In this context, ontologies can provide us with reference theories that articulate key domain distinctions; these distinctions are then reflected in modeling constructs, patterns and guidelines.

In this paper, we employ *ontological analysis* as a principled approach to assess capability modeling in ArchiMate. Our starting point is the ontology of capabilities described in [11]. This ontology is based on the Unified Foundational Ontology (UFO) [16] and is represented using the modeling language OntoUML [16]. It takes into account emergence [21, 31, 34, 40] and disposition theories [5, 14, 29, 32] to provide a well-founded conceptualization for capabilities and their relations.

The analysis presented in this paper builds up on our past work which proposes the representation of capability emergence in ArchiMate [9]. It also extends the work of Azevedo et al. [3, 4]. Although Azevedo and colleagues also analyzed ArchiMate’s support for capabilities using a UFO-based notion of disposition, they did not address detailed capability relationships. We argue that providing a proper account for these relationships is key to guide adequate capability representation in EA models and, consequently, for supporting the use of EA models in capability-based practices. To cite one example, this is needed to account for how organizational capabilities result from the relationships between the capabilities of various business entities (e.g., teams, departments, or other organizational structures.)

This paper is further structured as follows: Section 2 presents an overview of the literature related to capabilities and the Unified Foundational Ontology and the capability ontology; Section 3 presents the ontological analysis based on the capability ontology; Section 4 presents preliminary suggestions of enhancement to ArchiMate’s capability metamodel and specification which arise out of the ontological analysis; Section 5 discusses related work, and Section 6 concludes with our final remarks.

## 2 Background

Capability is generally defined as the “ability to do something” [30], a quality of being capable of achieving specific effects or declared objectives [38]. As noticed in most capability definitions, the meaning of *capability* is closely connected to the meaning of *ability*. Ability is commonly defined as the power to act, or as a kind of dispositional property that allows one to do something (useful or not) [23]. More specifically, the ability concept is defined by [23] as the “power that relates an agent to an action”. So, ability is a potentiality (power) related to a bearer and also to an action. What then distinguishes abilities in general from capabilities? A direct reference to value: ability definitions are usually more generic, and not directly related to “desired” results, outcomes, or achievements. They can be “value neutral” with respect to their associated

impact. Capabilities, in contrast, are conceived in terms of the usefulness of their outcomes and outputs. For example, in enterprise architecture and systems engineering, capabilities are defined as the “ability to do something useful” [24,25,28]; in information systems as the “ability to achieve a desired effect”; and, in the military field, it is defined as the “ability to achieve a determined military objective” [1]. Thus, a common tenet across these different areas is the reference to a “beneficial” aspect.

### 2.1 Modeling Capabilities in ArchiMate

ArchiMate is a widely used modeling language for EA. It provides a comprehensive framework for visualizing, analyzing, and communicating architectural blueprints within organizations. The language offers a standardized way to depict, understand, and manage the complexities of enterprise architectures. It serves as a bridge between business and IT domains, enabling stakeholders to align strategic goals with operational realities through a unified visual language [41].

The core of ArchiMate is its metamodel, which categorizes architectural elements into *behavior elements* and *structure elements* [41]. This dichotomy is depicted in the metamodel fragment shown in Figure 1. Behavior elements capture the dynamic aspects of an enterprise, such as processes, events, interactions, and other behaviors. Structure elements represent the static aspects, including organizational units, actors, roles, and equipment. ArchiMate divides structure elements into *active* structure elements and *passive* structure elements. So, it divides these elements in a way that is analogous to a division present in natural languages, with active structural elements like subjects, behavioral elements (like verbs), and passive structural elements (like objects) [41].

ArchiMate organizes architectural models into distinct *layers*, each representing a different perspective (or viewpoint) of the enterprise architecture. These layers include the *Business Layer*, focusing on organizational services, components, functions, and processes; the *Application Layer*, addressing software applications supporting business functions and services; and the *Technology Layer*, dealing with infrastructure and hardware components. ArchiMate also considers a perspective to represent motivational elements such as goals, drivers, requirements, and so on. In particular, the *Strategy Layer* focuses on business capabilities and resources [41] (which are shown as specializations of core elements in Figure 1).

Capabilities in ArchiMate’s metamodel are considered behavior elements (as events and processes) [41], in this case, *strategy behavior elements*. They represent an abil-

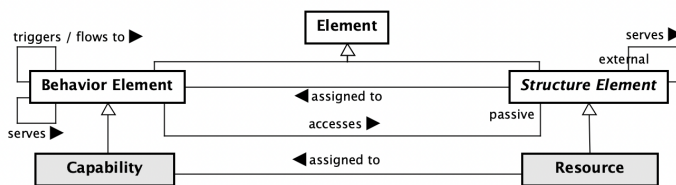
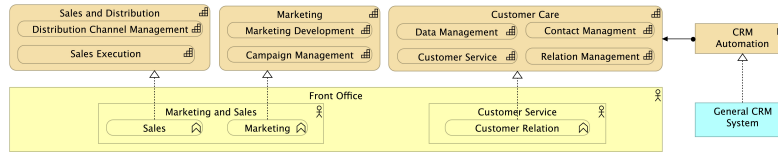


Fig. 1. Fragment of the ArchiMate’s Metamodel (adapted from [41])

ity that a Structure Element (organization unit, person, or system) possesses. Figure 2 illustrates an example of a capability model, corresponding to the ArchiSurance case study [19]. In ArchiMate models, they provide a high-level view of the current and desired abilities of an organization. As behavior elements, they can trigger, serve (contribute to), and flow (i.e., exchange matter, energy, or information) to each other (not considered in the figure). In the strategy layer, resources can be *assigned to* capabilities (e.g., the “CRM Automation” resource is assigned to the “Customer Care” capability). They can also be *realized by* other structure or behavior elements (e.g. business actors, business roles, business processes, business function, and so on). For example, the “Customer Care” capability in the example is *realized by* the “Customer Relation” function of the “Customer Service” actor. In this case, this means that these elements can be used to achieve a specific capability. Finally, capabilities can be related to other capabilities through *aggregation* and *composition*. In ArchiMate, “Composition is a whole/part relationship that expresses an existence dependency” [41] while aggregation is supposed to be a parthood relation that does not imply such a dependence. Whole/part relationships involving capabilities are depicted in Figure 2 by construct nesting: e.g., the “Marketing” capability encompasses “Marketing Development” and “Campaign Management”.



**Fig. 2.** Example of Capability Modeling in ArchiMate (extracted from [19])

## 2.2 Ontological Baseline

In order to account for capability-related phenomena more precisely, we employ here a fragment of the UFO foundational ontology [15]. UFO defines a system of domain-independent categories and their ties, which can be used to articulate conceptualizations of phenomena of interest. UFO has been developed based on theories from Formal Ontology, Philosophical Logics, Philosophy of Language, Linguistics, and Cognitive Psychology [16]. The employed fragment captures the first two basic ontological categories: that of the *types* (concepts, universals, e.g., Person, Planet, Music Band) and that of the *individuals* (particular things, e.g., John, Mary, Saturn, The Beatles). Individuals include *concrete individuals* and *abstract individuals*. *Concrete Individuals* are partitioned into *events* (a.k.a. *perdurants*), *endurants*, and *situations*. *Events* are *individuals* that occur in time, including processes, activities, actions, and tasks. Events are causally related and can be mereologically atomic or complex. Besides this, events can change, create, or terminate objects, including their aspects [17]. *Endurants* are individuals that persist in time possibly changing qualitatively while retaining their identity (i.e., people, organizations, cars). *Endurants* include *objects* and *aspects*. An *Object* is an endurant

that is considered existentially independent from other objects (like John, his car). Objects formed by parts (performing distinct functions) are called *functional complexes*. An *Aspect* is a reified property that *inheres in* an *endurant* (termed its bearer). Inherence is a type of existential dependence relation between aspects and their bearers. Aspects (as full-fledge *endurants*) have a lifecycle of their own and can be created, destroyed, or changed qualitatively in time while maintaining their identity.

Of special interest to us in this work is the UFO notion of *disposition*. *Dispositions* are *aspects* that can be manifested through the occurrence of *events* (possibly agents' actions, such as Anna's speaking English). In *situations* where *dispositions may manifest*, they are said to be "activated" (e.g., when a magnet is close to some ferrous material, or when Anna is prompted to introduce the topic of a meeting). Again, as *endurants*, they can themselves bear aspects, and change qualitatively through time [15].

In order to leverage UFO distinctions, we use conceptual modeling language OntoUML [16]. OntoUML is implemented as a UML profile that reflects the foundational distinctions and axiomatization put forth by UFO. Over the years, it has been used to model many core and domain reference ontologies in a variety of complex domains. We use it here to represent the capability ontology proposed in [9, 10]), on which we base our analysis (see Figure 3). As it is addressed, capabilities are considered special types of dispositions. They inhere in a (non-agentive or agentive) *capable object* (including functional complexes as systems). *Capabilities* are manifested through *capability manifestations*. Each capability manifestation is triggered by one or more *capability manifestation context* and brings about a *capability outcome* (both of which are situations). As depicted in this figure, the manifestation of a capability can also employ a *capability input* and produce (or change) a *capability output* (objects with a historical role). Capable objects participate in the capability manifestation with *enabler objects*, a role that represents objects needed for capable object manifestation (i.e., resources). Capabilities can be *interactive*, when they play a 'role' in certain relations to each other (and other dispositions). The interaction relation, in this case, corresponds to relations that include relations of reciprocity [14, 32], enabling [5], and changing [32], among others (not all depicted in the figure). We use the term "interacts with" as an abstract supertype to these relations. *Capabilities* can also be complex or atomic. *Complex capabilities* are those that have other *capabilities* (and dispositions) as proper parts, and *atomic capabilities* are those capabilities that are not complex. *Capabilities* inhere in *objects* (capable

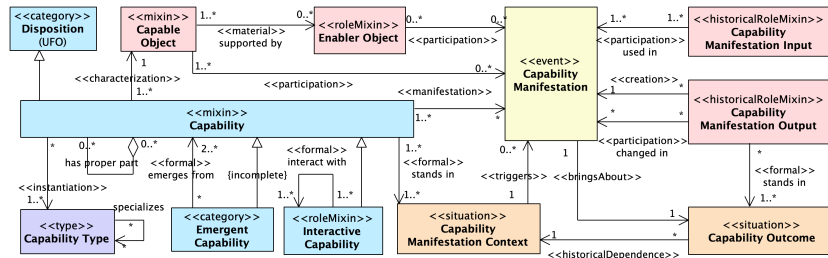


Fig. 3. Fragment of the Disposition and Capability Ontology used in the ontological analysis

objects), which can be atomic objects or functional complexes (including systems). The *capabilities* of capable objects that are complex (functional complexes as systems) can be *emergent capabilities* or *resultant capabilities* [10]. *Resultant capabilities* are those that come “directly” from some *particular dispositions* of components (i.e., component capabilities). They can even be present in system parts in isolation. For example, the capability of a car to play music is a direct result of the capability of the car’s stereo to play music. In contrast, *emergent dispositions* are those that, while related to the (interactive) dispositions of system parts, are not present in isolation in the separated parts [8]. For example, the “buoyancy” of a steel ship is not present in an arbitrary piece of its hull in isolation. Finally, capabilities instantiate one or more capability types. As depicted, a capability can specialize another.

### 3 Ontological Analysis of Capability-Related Elements in ArchiMate

As approached, the phenomena of capability emergence and interaction face challenging issues of capability modeling in EA modeling notations, especially regarding ArchiMate. As a result, these aspects motivate us to employ *ontological analysis* [44] as a principled approach to evaluate and identify ontological deficiencies concerning capability modeling in ArchiMate. This ontological analysis is based on the capability ontology [11] fragment briefly presented above. This principled approach is fundamental to prevent ontological deficiencies such as semantic overload, construct redundancy, construct deficit, and construct excess [44]. *Construct excess* occurs when a notation construct does not align with any ontological concept; *construct overload* happens when a single notation construct can represent several ontological concepts; *construct redundancy* arises when multiple modeling elements can represent a single ontological concept; and *construct deficit* exists when there is no notation construct in the modeling language for a specific ontological concept.

Our approach involves identifying and examining the fundamental capability-related elements and their relationships within ArchiMate’s metamodel. We delve into the descriptions and meanings of these elements as specified in ArchiMate’s specification, aiming to understand their relevance in enterprise architecture. Then, we compare ArchiMate’s capability-related elements with capability ontology concepts, particularly focusing on UFO distinctions. This comparison aims to establish correspondences and clarify how ArchiMate’s capability modeling aligns with broader ontological frameworks. Next, we present the ontological analysis and, based on that, we identify some ontological deficiencies in ArchiMate. For each of them, we provide the related context and problems, explaining how this impacts the quality of ArchiMate models.

#### 3.1 Capabilities as Aspects

As presented before, capability is defined in distinct fields as an *ability* of a certain entity (a system, organization, enterprise, person, etc.) to do something and generate some outcome. This ability corresponds to a “power”, “propensity”, or “potentiality” of the bearer, i.e., a potential to act or behave. Then, when an activating situation happens,

the capability manifests itself as an event or behavior. The literature, mainly related to disposition theories, makes a clear distinction between the capability (as a potentiality) and its manifestation (the event). In a similar sense, the definition of capabilities in ArchiMate states that they are abilities owned by organizations, people, or systems. Additionally, as presented in ArchiMate’s metamodel, capabilities are categorized as *behavior elements*, similar to events and processes, related to the organization’s dynamics (action, events, etc).

*Ontological analysis of capability meaning:* For UFO, aspects are intrinsic characteristics of *endurants* (especially objects). Moreover, it takes dispositions to be a subtype of aspects that are manifested in certain situations through events. In addition, according to the capability ontology (and also the ontological analysis performed in [3]), capabilities are dispositions that characterize *capable objects* and are manifested by events called *capability manifestation*. So, according to the ontology, capabilities are not behavior (event) itself; instead, they are aspects (dispositions) inhering in (capable) objects that possibly manifested through events (behavior).

*Diagnosis of ArchiMate’s worldview:* The notion of capability in ArchiMate is at odds with the understanding of capability as a “potentiality”, i.e., a potential behavior, not the behavior itself. Furthermore, the ArchiMate metamodel does not include elements related to “aspect” or “characteristic”. In line with [44], this is a case of *construct deficit*, since there is no construct representing capabilities itself (as an aspect) but just their manifestation. As a result, there is no construct to represent capable objects and the inherence relation binding an aspect to its bearer. In summary, ArchiMate’s conception of capability as a behavioral element is inadequate because capabilities are aspects of structural elements that manifest themselves as behavior. As a consequence of this ontological deficiency, we have two main *problems*: (i) relations that would usually apply to behavior elements (such as triggers, flow, etc.) should not be applicable to capabilities or overlap with other relations involving capabilities; (ii) ArchiMate has no specific construct to identify the capable object that bears a particular capability. The ArchiMate relation of assignment could in principle be used if one could semantically overload it to represent the relation of inherence.

### 3.2 Capability Composition and Decomposition

When modeling capabilities in activities such as capability mapping, these are structured hierarchically in order to better support their understanding, as well as of their role in gap analysis [20, 36]. For example, in Figure 2, “Sales and Distribution” is decomposed in “Distribution Channel Management” and “Sales Execution”.

*Ontological analysis of capability decomposition:* There are multiple ways a capability can be related to its parts. Firstly, we can have direct capabilities of an entity that are decomposable into simpler capabilities inhering in the very same entity. Secondly, we can have capability emergence, in which case a capability of the whole emerges from the interaction of more basic capabilities inhering in its parts (e.g., as it happens with capabilities such as resilience, adaptability, innovation, etc). In this latter case,

we must also consider the particular structure relating these parts and affording the interaction between their capabilities. For example, how certain capabilities of a entire organization emerge from the interaction between the capabilities of organizational units, teams, or employees. In other words, in the case of emergence, we have more than just a mereological relation between wholes and parts, but we need to capture the structural unifying relations between the composing capabilities themselves. Finally, we have the case of resultant capabilities, in which a capability inhering in the whole is directly derived from a capability inhering in one of its parts. In summary, according to the reference ontology employed here, capability decomposition can have two distinct meanings: (i) mere mereologically complex capabilities: when a complex capability is just split into simple capabilities independently of the structure of the capable object; and, (ii) emergent and result capabilities: when a capability of a capable object emerges or results from interactive capabilities inhering in its parts.

*Diagnosis of ArchiMate's world view:* ArchiMate does not identify the bearer of a capability and hence it does not distinguish *capability decomposition* from *capability emergence*. As such, it only allows for the use of the simple mereological relations of aggregation and composition between capabilities. We have then a case here of *construct deficit* [44]. Because of this ontological deficiency, we have the following *problems*: (i) a significant gap between the capabilities modeled at the strategy layer and their implementation in the business, application, and technology layers (since capabilities of the whole are not related to the capabilities of part); (ii) difficulty in understanding (and replicate) patterns and best practices that promote the emergence of desirable capabilities due to not having the phenomenon of emergence in focus.

### 3.3 Interactions Between Capabilities

Capabilities are not isolated entities since they can be combined, impact, and collaborate with each other. Sometimes, they depend on each other so that can be enacted to satisfy certain goals. For example, in Figure 2, “Campaign Management” needs the outcomes from “Marketing Development”; the same happens between “Relation Management” and “Contact Management”.

In the context of EA, it is important to identify how capabilities impact enterprise elements, especially other capabilities [42, 43]. For instance, initiatives to increase the innovation capability in organization can generate unexpected side effects. This, in the end, can harm other capabilities from a global perspective of the organization, as [39] argues about organizational learning. A similar issue happens with technical and social capabilities in organizations as socio-technical systems [2]. Capability interaction is related to capability emergence as previously discussed, as it also involves understanding how capabilities emerge in the whole organization; however, here, we focus not on the hierarchical perspective of capabilities, but with how they relate “horizontally”.

ArchiMate allows the representation of dynamic and dependency relationships between capabilities. Capabilities can *serve*, *trigger*, or *flow* to other capabilities. The *serves* relationship is a dependency relationship that indicates a “contribution” from one capability to another. The *triggering* relationship represents a temporal or “causal precedence” between capabilities. The *flowing* relationship is also dynamic, corresponding

to an “exchange” of matter, energy, or information between capabilities. Besides being present in the notation, these relationships are not as commonly used as the hierarchical decomposition in capability models, as evidenced in [19].

*Ontological analysis of capability relationships:* Analyzing the dynamic relationships (flowing and triggering) from an ontological point of view, we understand that capabilities can interact indirectly through their manifestations: the *flowing relationship* means that capability  $c_1$  manifests through an event  $e_1$ , producing output  $o_1$ . After this, capability  $c_2$  manifests through event  $e_2$ , using output  $o_1$  as input. Thus, it is actually the manifestation  $e_1$  that “flows to” manifestation  $e_2$ ; in the *triggering relationship*, capability  $c_1$  triggering capability  $c_2$  means that  $c_1$  manifests through event  $e_1$ , generating an outcome  $o_1$ . This outcome activates capability  $c_2$ , triggering its manifestation through event  $e_2$ .

In the case of the *servicing relationship*, the term “contribute” - used by ArchiMate to characterize this relation, can have various interpretations. Depending on the interpretation, there can even be an overlap between the servicing relationship and other relationships. First, servicing could mean that capability  $c_1$  serves capability  $c_2$  if  $c_1$  provides something (matter, information, or energy) to  $c_2$ . However, this overlaps with the flowing relationship, where one capability provides something to another. Second, servicing could mean “contributing to the manifestation”, where  $c_1$  serves  $c_2$  if  $c_1$  contributes to the manifestation of  $c_2$ . This interpretation also overlaps with the triggering relationship, as it represents the creation of conditions for the manifestation of another capability. Third, a similar interpretation involves reciprocity. Ontologically speaking, reciprocal capabilities need each other to manifest together (e.g., “Product Purchasing” and “Product Selling”). Here,  $c_1$  serves  $c_2$  if they are reciprocal and contribute to each other’s manifestation. Fourth, servicing could mean “additionality”, i.e., capabilities  $c_1$  and  $c_2$  manifesting through the events  $e_1$  and  $e_2$  that contribute to the same outcome  $o_1$ . For example, the capability of software testing contributes to the software coding capability by improving the quality of the software coding output. In the capability ontology, these capabilities are called “additional capabilities”. Fifth, servicing could mean that  $c_1$  contributes to the development (improvement) of  $c_2$ . Ontologically speaking, a capability can change another capability or change the conditions for the latter’s manifestations. For instance, the manifestation event  $e_1$  of capability  $c_1$  changes a quality of capability  $c_2$  (a direct change) or change the capability bearer  $b_1$  of  $c_2$  in a way that it changes the possible manifestations of  $c_2$  (e.g., the technological learning capability that makes the development of software development capability possible).

*Diagnosis of ArchiMate’s worldview:* The conception of capability relations in ArchiMate do not fully consider the nature of capabilities, leading to both cases of *construct overload* and also *construct excess* [44]. This is due to ArchiMate’s misguided conception of these relationships purely from the perspective of behavioral elements, as if capabilities were directly comparable to events and processes that have temporal precedence and that can exchange matter, energy, or information. As we have discussed before, capabilities are to be distinguished from their manifestations. Besides this, the ontology also provides further distinction among relationships that can obtain between capabilities (e.g., blocking or disabling) and which are not addressed by ArchiMate, i.e.,

a case of *construct deficit* [44]. In summary, we conclude that ArchiMate’s conception of capability as a behavioral element prevents important (“horizontal” capability) relations from being identified and properly expressed in the language. As a result of these ontological deficiencies, we have trouble: analyzing capability emergence - since these relations are key to explaining that phenomenon; planning and prioritizing capability development - since we cannot properly identify and express enabling and supporting capabilities); identifying negative capability interactions (capabilities that harm others).

### 3.4 Capabilities and Resources Assignment

Organizations are socio-technical systems that encompass distinct kinds of resources, from people to equipment such as hardware, software, tools, and even the physical environments in which these systems are situated [2, 12, 27]. The interaction between resources must occur appropriately for the organization to develop its desirable capabilities. However, understanding how resources interact can be an intricate issue given the different natures of these different kinds of resources (especially human and technical ones) [22]. For example, it is common to see in organizations the belief that the acquisition of technical resources (e.g., a new software system) will be sufficient for developing a desired capability. However, in this example, if people do not have the correct skills to properly use such a technology or have a dissonant attitude towards it (e.g., resistance to that technology), such investment will be fruitless [2, 13]. So, the acquisition of new resources does not necessarily imply improvements in the people and organization as a whole [2, 7, 12].

In ArchiMate, resources are structure elements that belong to the strategy layer. According to the specification, resources include *tangible assets* (i.e., physical or financial assets), *intangible assets* (i.e., technological, reputational, or cultural assets), and *human assets* (e.g., skills, knowledge, or know-how). According to the ArchiMate metamodel, resources can be assigned to capabilities. The assignment relationship relates structural elements to behavior elements, indicating the “allocation” of structure elements to a behavior. In the business layer, this relationship is typically used to link business actors to a business role or to a business process they are supposed to perform.

*Ontological analysis of the resource assignment to capabilities:* Tangible resources can be considered objects in UFO. In the capability ontology, capabilities are tied to a *capable object* (the bearer of that capability). These capable objects can also be linked to *enabler objects*, which are required for the manifestation of a capability. In ArchiMate, resources are assets owned or controlled by a structural entity, something which is to be expected in this context also for enabler objects. So, we consider resources both as enabler objects, as well as capable objects - since they possess capabilities that interact with the ones from other capable objects. Thus, based on these ontological distinctions, the assignment of a resource to a capability can have two meanings: (i) a resource  $r_1$  is assigned to capability  $c_1$  if it corresponds to a *capable object* characterized by a capability  $c_2$  of same type of  $c_1$  (e.g., the software developer human resource assigned to the software development capability); (ii) a resource  $r_1$  is assigned to capability  $c_1$  if it is an *enabler object* characterized by a capability  $c_2$  needed for the manifestation of capability  $c_1$  (e.g., the laptop equipment assigned to the software development capability).

This understanding aligns with the perspective presented in [3]. There, in order to be assigned to a capability, a (tangible) resources must have dispositions aligned with that capability. However, as the authors note there, this is not necessarily a one-to-one mapping since multiple resources can be assigned to the same capability. Here again, we have two ways of interpreting this assignment relationship: one related to instantiation and the other to emergence. The first case happens when individual resources (objects)  $o_1, o_2, o_n$ , characterized by individual capabilities as  $c_1, c_2, c_n$ , are assigned to a capability type  $C$ . In this case, the capabilities  $c_1, c_2, c_n$  are instances of capability type  $C$ . An example is the assignment of the resources ‘iOS dev. team’ and ‘Android dev. team’ to the software development capability (type): both teams are individuals characterized by their respective individual software development capabilities. The second case happens when resources assigned to a capability  $c$  are objects  $o_1, o_2, \dots, o_n$  characterized by respective capabilities  $c_1, c_2, \dots, c_n$ . These capabilities interact with each other (i.e., reciprocal, additional, enabling, changing, and so on), and as a result, they are responsible for the emergence of that capability  $c$ . For example, the *software development capability* of a company emerges from the combination of various human resources (front-end developer, back-end developer, tester) as well as other tangible resources, e.g., hardware and software resources.

To further deepen this analysis, we can consider the notion of functional complexes from UFO [15] and the distinctions put forth by the system ontology in [10]. In these ontologies, organizations are seen as special types of systems (or functional complexes in UFO) being composed of interconnected functional parts that play functional roles w.r.t. to the whole organization. Thus, an “assignment” of a resource to an organization’s capability can be understood as its allocation to a *functional role* in that *organization system* (or subsystem) in order to contribute to the achievement of the referred capability. More generally, the assignment of resources  $r_1, \dots, r_n$  to capability  $c$  in organization  $o_1$  is meant to signify that those resources playing the functional roles  $f_1, \dots, f_n$  inside  $o_1$  are necessary *components* of that organization (as a system) needed to achieve that capability. In the example of the “software development capability”, behind all distinct human resources assigned to this capability there is a (social) subsystem of the organization – a team – composed of front-end developers, back-end developers, etc., and which collaborates with the “software development capability” achievement. In this case, particularly, the assignment of these human resources to this capability means that they are allocated to a team having that capability.

*Diagnosis of ArchiMate’s worldview:* ArchiMate lacks a clear semantics for the assignment relationship, and the notation itself does not provide a well-defined meaning for the resource construct. Resources can represent both physical objects (tangible resources) as well as intangible entities like cultural aspects, values, etc. As resources are taken to be structural entities, this implies that intangible assets (such as skill or competences) can also be structural elements, i.e., parts of organizational structure in a way analogous to organizational units. Furthermore, ArchiMate does not allow for explicitly representing that a resource bears a certain capability (a case of construct deficit [44]) but only generically that resources are assigned to capability, without clear guidelines on how these assignments should be made. However, to effectively combine resources in order to achieve a certain capability, it is necessary to understand their nature,

their relations, and also their dispositions (including capabilities and vulnerabilities). We also have again here a case of *construct overload* [44], in which several interpretations can be associated to the assignment relationship: the resource is a capable object that bears that capability; or it is a part of the organization as a capable object; or an enabler object required in capability manifestations (hence also a bearer of capabilities). As a result of these deficiencies, ArchiMate lacks expressivity to properly represent and analyze resources, their interactions, and related capabilities, which also impacts the analysis and representation of capability interaction and emergence.

### 3.5 Capability Implementation

In the EA context, capability can be approached from a strategic perspective and also from a more operational perspective. In the strategic perspective, capabilities are seen in a more abstract way, independent of their possible implementation. In this case, capability models are more future-driven, focusing on the desired capabilities of the organization. On the other hand, from the operational perspective of capabilities, they are represented as they are in the present, considering the actual stage of the organization. The focus is more on the present and on the “how”, not just on the “what”. From the operational perspective, understanding how capabilities manifest is important to understand how they work in practice and how to implement them. In this case, one needs to understand not only the manifestation of a focal capability itself but also how the manifestation of how distinct capabilities are linked and also how the results and outcomes of these capability manifestations are related.

Concerning the strategy and operation perspectives, ArchiMate distinguishes between the strategy and other layers (business, application, and technological layers). The strategy layer is more abstract and implementation-independent, while the others are more specific, focusing on the present and how the organization is implemented [42]. In ArchiMate, capabilities are represented especially in the strategy layer and are realized by elements in the “lower” layers, such as the business layer. Thus, capabilities represented in the strategy layer are implemented by business functions, business processes, business actors, business roles, and also elements from the application and technology layers. This *realization* relationship indicates that more abstract elements (focused on “what”) are realized by more tangible elements (“how”). Thus, an “abstract” capability can be implemented by various elements combined in the business layer, such as business actors, roles, services, processes, events, and so on.

*Analysis of Capability Implementation:* Capabilities are individuals that instantiate *capability types*, i.e., that are classified by them. As detailed in the ontology, a capability type can specialize others. The ontology allows for considering different levels of abstraction between capabilities. With this distinction in mind, we consider that *strategic capabilities* of an organization are “more abstract” capabilities that focus on the “what” and can be specialized in various ways depending on the “how”. *Operational capabilities* are specializations of strategic capabilities that concern a specific way to realize them, making them more concrete and relating them to a specific implementation within the organization. For example, suppose that *software development capability* is a strategic capability for a given organization. In this case, it could be specialized into different

operational capabilities such as *web system development* or *mobile app development*, which could further specialize into *iOS app development* and *Android app development*.

Regarding the realization of capabilities by structural elements from “lower” layers (i.e., business, application, and technology), we encounter a situation similar to what we saw with the *assignment relationship* between capabilities and resources. Similarly, we can interpret that structural elements are capable objects characterized by capabilities and which can realize (strategic) capabilities as a consequence. But, the realization of a capability by a structural element can also have distinct meanings, such as: (i) the structural element  $s$  that realizes a strategic capability  $c$  is a *capable object* characterized by  $c'$  - an *operational capability* that specializes  $c$ ; (ii) the structural element  $s$  that realizes a strategic capability  $c$  is an “enabler” object necessary for a capable object  $o$  to manifest  $c'$  - an operational capability that specializes  $c$ . If a strategic capability is realized by multiple structural elements, similarly to resource assignment, we have that: the set of structural elements  $s_1, s_2, \dots, s_n$  are characterized by operational capabilities  $c_1, c_2, \dots, c_n$ , and that these capabilities, when combined (through interaction relationships), contribute to the emergence of  $c'$  - an operational capability that specializes the strategic capability  $c$ . On the other hand, if a capability  $c$  is realized by a *behavior element*  $b$ , this means that: (i) there is a structural element  $s$  characterized by the operational capability  $c'$ , which specializes  $c$  and manifests through  $b$ ; (ii) Or, there is a set of structural elements  $s_1, s_2, \dots, s_n$ , characterized by operational capabilities  $c_1, c_2, \dots, c_n$ , which, when combined (through interaction relationships), contribute to the emergence of  $c'$  - an operational capability that specializes  $c$  and manifests through  $b$ .

*Diagnosis of ArchiMate’s worldview:* ArchiMate does not provide an appropriate representation for the implementation of capabilities. As we saw, structural elements at different levels, such as strategic, business, application, and technology, can also be capable objects characterized by capabilities at different levels. However, ArchiMate does not allow for the representation of the structural elements as bearers of capabilities in any of the layers considered in the notation. This makes it difficult to map strategic-level capabilities to structural elements at other levels, as well as to relate structural elements in a way that combines their capabilities (through interaction relationships) to collectively generate the desired capability at the strategic level. Besides this, the notation only allows for the representation of capabilities in the strategic layer, leaving no construct in the business layer to represent “operational” capabilities. In addition, ArchiMate’s *realization relationship* is not precisely defined, being another case of *construct overload* [44] as detailed in the ontological analysis.

Capabilities can be *realized* by both structural and behavioral elements, which can lead to many misunderstandings. There are no guidelines on how to combine these elements in order to implement capabilities. In summary, ArchiMate’s capabilities are only considered as “abstract” entities in the strategy layer. Capabilities are not addressed in other layers, in which they are often treated as (but not properly distinguished from) functions. Structural elements in business, application, and technological layers are also capable objects characterized by capabilities, which can also engage in fruitful interactions.

## 4 Preliminary Suggestions to ArchiMate Enhancement Based on the Ontological Analysis

Based on the ontological analysis we have presented thus far, in this section, we consider some preliminary suggestions to improve the representation of capabilities and their relationships in ArchiMate. These include the suggestion of a few capability-focused viewpoints and the identification of possible revision of language constructs related to capabilities.

*Capability-Focused Viewpoints:* Regarding capability (de)composition, it would be beneficial to distinguish between complex and emergent capabilities in the hierarchical representation used in capability mapping. Since the strategy view does not allow for the representation of the bearer of capabilities, it would be helpful to represent different levels of organizational granularity. At least three levels could be represented: the organizational level, the organizational unit level (e.g., teams, departments), and the individual level. Capabilities at each level could be then be visually distinguished.

Concerning capability interaction, it would be useful to have guidelines to model capability interaction, focusing on the interaction between capabilities that belong to the same level in the organization. This would help better understand the emergence of capabilities, which arise from their interactions. Additionally, it is important to clarify the semantics of capability relationships, as their current descriptions are generic and similar to other relationships between structural elements, which have a different nature. Specializing the semantics of the *servicing* relationship between capabilities is recommended, including all possibilities such as reciprocity, additionality, and change. The *servicing* relationship description would also make clear how it differs from the *flowing* and *triggering* relationships. Additionally, the language should provide better guidelines on how to implement capabilities in the resources they are assigned to, ensuring these capabilities are realized by structural elements interconnected in a proper way. A notion such as functional complex (from UFO) is essential in this context.

One way to assist in modeling the aspects mentioned above is through viewpoints, as they can assist modelers by providing perspectives based on the aforementioned aspects. For example, it would be key to have such a support for modeling emergence and interaction between capabilities, as these aspects are closely related to others, such as the assignment of resources and the implementation of capabilities. These viewpoints would offer a clearer way to visualize and manage how capabilities emerge and interact within an organizational structure.

*Language Constructs Revision:* Regarding the representation of capabilities, a general suggestion is to consider adding an element in the metamodel related to ‘aspects’ in order to distinguish capabilities from behavioral elements. An option in this case is to refactor the metamodel and possibly introduce a separate notion of “dependent” structural element, and in any case, a relation to establish the active structure element who bears an aspect (capabilities included). In this proposed redefinition, it is also important to specialize the relationships between capabilities based on this new understanding of capability (as an aspect), rather than simply inheriting the meanings of relationships from behavior entities. Additionally, a relationship that can be included in this refactoring is

the *characterization relationship*, linking active structure elements (the bearers) to their respective capabilities. Finally, considering these possible changes, it would be valuable to include capabilities in all layers, similar to how ArchiMate handles structural and behavioral elements (e.g., business function, application function, technical function, etc.). In this case, there would be different types of capabilities based on the layer, such as strategy capability, business capability, and application capability, and one could also express that capabilities from higher layers are realized by capabilities from lower layers. An option, in this case, might be to use *business function* elements to represent “operational” capabilities (see Figure 2). However, this requires some clarification on the interpretation of functions in ArchiMate..

## 5 Related Work

Other related works that employ Foundational Ontologies in EA modeling include [4, 33, 35, 37]. Azevedo et al. [4] perform an ontological analysis of capability, resource, and competence. The authors discuss especially the definition of capability based on UFO; we adopt and build up on that analysis in the present work. As an application, the authors propose improvements in Enterprise Modeling (using ArchiMate), through a metamodel connecting capabilities and the strategy layer with motivational aspects. Capabilities can be aggregated (with resources) in what the authors call “capability bundles” [3]. The work proposed by Nardi et al. [33] focuses on the ontological analysis and modeling of the Service concept. In a complementary way, [33] states that one dimension of Service Modeling is to represent a manifestation of capabilities. However, although both papers address the subject of capabilities using UFO, they do not delve into this topic since this is not the focus of both papers. Sales et al. [37] proposed improvements in the ArchiMate notation based on an ontological analysis, focusing on the concept of value. In their analysis, they included the concept of capability, given its strong relation to value. According to their interpretation, and as adopted here, capabilities are dispositions with valuable impacts on a value subject. However, fully characterizing is not the main focus of their work. Building on the work in the value domain proposed in Sales et al. [37], Oliveira et al. [35] conduct an ontological analysis of ArchiMate focused on security and propose a redesign of the language. In this security-focused analysis, they considered other types of dispositions, such as vulnerabilities, corresponding to those with undesired effects. Additionally, the authors extend the meaning of capabilities to include *threat capabilities*. Although these related works address capabilities in the context of EA, given their respective foci, none of them all the subtle phenomena related to capability modeling.

## 6 Final Remarks

In this work, we presented an ontological analysis of ArchiMate based on a well-founded capability ontology. To perform this analysis, we first identified several issues concerning the semantics of the ArchiMate metamodel, especially regarding capability emergence and capability interaction. We also uncovered semantic issues related to the relationship between capabilities and resources, as well as capability implementation. This work

may serve as a foundation for proposing language redesign, language patterns, or even language extensions. The issues identified can be a starting point for the proposal of new capability representations. This work not only contributes to enhancing ArchiMate's capability modeling but also impacts capability modeling in general, potentially influencing other EA notations. For example, in future work, we intend to perform a similar ontological analysis of the Unified Architectural Framework [26].

**Acknowledgments.** This study was financed in part by the Brazilian funding agencies CNPq (443130/2023-0, 313412/2023-5) and FAPES (2021-GL60J, 2022-NGKM5, 1022/2022).

## References

1. Antunes, G., Borbinha, J.: Capabilities in systems engineering: an overview. In: Exploring Services Science: 4th International Conference, IESS 2013, Porto, Portugal, February 7-8, 2013. Proceedings 4. pp. 29–42. Springer (2013)
2. Appelbaum, S.H.: Socio-technical systems theory: an intervention strategy for organizational development. *Management decision* **35**(6), 452–463 (1997)
3. Azevedo, C.L.B., et al.: An Ontology-Based Well-Founded Proposal for Modeling Resources and Capabilities in ArchiMate. In: 17th IEEE International EDOC Conference (EDOC 2013). pp. 39–48. IEEE Computer Society Press (2013)
4. Azevedo, C.L.B., Iacob, M., Almeida, J.P.A., van Sinderen, M., Pires, L.F., Guizzardi, G.: Modeling resources and capabilities in enterprise architecture: A well-founded ontology-based proposal for ArchiMate. *Information Systems* **54**, 235–262 (2015). <https://doi.org/10.1016/j.is.2015.04.008>
5. Barton, A., et al.: A taxonomy of disposition-parthood. In: Workshop on Foundational Ontology in Joint Ontology Workshops: JOWO 2017. vol. 2050, pp. 1–10. CEUR-WS: Workshop proceedings (2017)
6. Bernus, P.: Enterprise models for enterprise architecture and ISO9000:2000. *Annual Reviews in Control* **27**(2), 211–220 (Jan 2003)
7. Bruseberg, A., Lintern, G.: Human factors integration for modaf: Needs and solution approaches. In: INCOSE International Symposium. vol. 17, pp. 1240–1255. Wiley Online Library (2007)
8. Bunge, M.: *Treatise on Basic Philosophy. Ontology II: A World of Systems*. Springer Netherlands, Dordrecht, Netherlands (1979)
9. Calhau, R.F., Almeida, J.P.A., Kokkula, S., Guizzardi, G.: Modeling competences in enterprise architecture: from knowledge, skills, and attitudes to organizational capabilities. *Software and Systems Modeling* pp. 1–40 (2024)
10. Calhau, R.F., Prince Sales, T., Oliveira, Í., Kokkula, S., Ferreira Pires, L., Cameron, D., Guizzardi, G., Almeida, J.P.A.: A system core ontology for capability emergence modeling. In: International Conference on Enterprise Design, Operations, and Computing. pp. 3–20. Springer (2023)
11. Calhau, R.F., Almeida, J.P.A., Guizzardi, G.: A capability reference ontology for the analysis and design of system capabilities, <https://purl.org/nemo/paper/creon>, pre-print
12. Cummings, T.G.: Self-regulating work groups: A socio-technical synthesis. *Academy of management Review* **3**(3), 625–634 (1978)
13. Ellen, P.S., Bearden, W.O., Sharma, S.: Resistance to technological innovations: an examination of the role of self-efficacy and performance satisfaction. *Journal of the academy of marketing science* **19**, 297–307 (1991)

14. Galton, A., et al.: Dispositions and the infectious disease ontology. In: *Formal Ontology in Information Systems: Proceedings of the Sixth International Conference (FOIS 2010)*. vol. 209, p. 400. Ios Press (2010)
15. Guizzardi, G., Wagner, G., Almeida, J.P.A., Guizzardi, R.S.S.: Towards ontological foundations for conceptual modeling: The unified foundational ontology (UFO) story. *Applied Ontology (Online)* **10**, 259–271 (2015). <https://doi.org/10.3233/AO-150157>
16. Guizzardi, G.: *Ontological Foundations for Structural Conceptual Models*. No. 15 in *Telematica Inst. Fundamental Research Series*, Telematica Inst., Enschede, The Netherlands (2005)
17. Guizzardi, G., et al.: Towards ontological foundations for the conceptual modeling of events. In: *Conceptual Modeling*, pp. 327–341. Springer Berlin Heidelberg (2013)
18. International Organization for Standardization (ISO): *ISO/IEC/IEEE 42020:2019 Software, Systems and Enterprise–Architecture Processes* (2019)
19. Jonkers, H., Band, I., Quartel, D., Lankhorst, M.: *Archisurance case study (version 3.1)*. Tech. rep., The Open Group (2019)
20. Josey, A.: *TOGAF® version 9.1-A pocket guide*. Van Haren (2016)
21. Juarrero, A.: Dynamics in action: Intentional behavior as a complex system. *Emergence* **2**(2), 24–57 (2000)
22. Kochan, T., Cutcher-Gershenfeld, J.: *Integrating social and technical systems: lessons from the auto industry* (2008)
23. Maier, J.: Abilities. In: Zalta, E.N., Nodelman, U. (eds.) *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2022 edn. (2022)
24. Martin, J., Fairley, D., Lawson, B., Faisandier, A.: Enterprise systems engineering background. In: *Guide to the System Engineering Book of Knowledge (SEBoK)*, version 2.5, pp. 642–650. BKCASE (2021), <https://sebokwiki.org/w/images/sebokwiki-farm!w/2/24/SEBoKv2.5.pdf>
25. Martin, J., Lawson, B., Faisandier, A.: Enterprise capability management. In: *Guide to the System Engineering Book of Knowledge (SEBoK)*, version 2.5. BKCASE (2021), <https://sebokwiki.org/w/images/sebokwiki-farm!w/2/24/SEBoKv2.5.pdf>
26. Martin, J.N., O’Neil, D.P.: Enterprise architecture guide for the unified architecture framework (uaf). In: *INCOSE International Symposium*. vol. 31, pp. 242–263. Wiley Online Library (2021)
27. Mavor, A.S., Pew, R.W. (eds.): *Human-system integration in the system development process: A new look*. The National Academies Press (2007)
28. Merrell, E., et al.: *Capabilities* (2022)
29. Molnar, G., Bradley, N.: *Powers: A study in metaphysics*. Clarendon Press (2003)
30. Morgan, P.: The concept of capacity (draft version). *Study on capacity, change and performance* pp. 1–19 (2006)
31. Mossio, M., et al.: Emergence, closure and inter-level causation in biological systems. *Erkenntnis* **78**(S2), 153–178 (2013)
32. Mumford, S., Anjum, R.: *Getting Causes from Powers*. Oxford University Press (2011)
33. Nardi, J., Falbo, R., Almeida, J., Guizzardi, G., Ferreira Pires, L., van Sinderen, M.J., Guarino, N., Fonseca, C.: A commitment-based reference ontology for services. *Information Systems (Oxford)* **54**, 263–288 (2015). <https://doi.org/10.1016/j.is.2015.01.012>
34. O’Connor, T.: Emergent properties. *American Philosophical Quarterly* **31**(2), 91–104 (1994)
35. Oliveira, Í., Sales, T.P., Almeida, J.P.A., Baratella, R., Fumagalli, M., Guizzardi, G.: Ontological analysis and redesign of security modeling in archimate. In: *IFIP Working Conference on the Practice of Enterprise Modeling*. pp. 82–98. Springer (2022)
36. Open Group (Reading, E.): *ArchiMate® 3.0. 1 Specification: The Open Group Standard*. Van Haren Publishing (2017)

37. Sales, T.P., Roelens, B., Poels, G., Guizzardi, G., Guarino, N., Mylopoulos, J.: A pattern language for value modeling in archimate. In: *Advanced Information Systems Engineering: 31st International Conference, CAiSE 2019, Rome, Italy, June 3–7, 2019, Proceedings 31*. pp. 230–245. Springer (2019)
38. Saxena, M.: *Capability Management*. Global India Publications (2009)
39. Senge, P.M.: *The fifth discipline. Measuring Business Excellence* (1997)
40. Spencer-Smith, R.: Reductionism and emergent properties. In: *Proceedings of the Aristotelian Society*. pp. 113–129. JSTOR (1995)
41. The Open Group: Archimate 3.2 specification (2023), <https://pubs.opengroup.org/architecture/archimate3-doc/>
42. The Open Group: TOGAF Business Capabilities Guide V2. <https://pubs.opengroup.org/togaf-standard/business-architecture/business-capabilities.html> (2023), [Accessed 06-10-2023]
43. Thorn, S.: Redefining traceability in Enterprise Architecture and implementing the concept with TOGAF 9.1 and/or ArchiMate 2.0 (2013), <https://blog.opengroup.org/author/opengroupblog/>
44. Weber, R.: *Ontological foundations of information systems*. (No Title) (1997)