Service Commitments and Capabilities Across the ArchiMate Architectural Layers

Conference Paper · September 2016
DOI: 10.1109/EDOCW.2016.7584386

7 authors, including:

Julio Cesar Nardi
Federal Institute of Espírito Santo (Ifes)
26 PUBLICATIONS **181** CITATIONS
SEE PROFILE

João Paulo A. Almeida
Universidade Federal do Espírito Santo
148 PUBLICATIONS **1,549** CITATIONS
SEE PROFILE

Ricardo de Almeida Falbo
Universidade Federal do Espírito Santo
172 PUBLICATIONS **1,661** CITATIONS
SEE PROFILE

Maria-Eugenia Iacob
University of Twente
114 PUBLICATIONS **1,275** CITATIONS
SEE PROFILE

Service Commitments and Capabilities Across the ArchiMate Architectural Layers

Julio Cesar Nardi[1], João Paulo A. Almeida[2], Maiara Candido Pereira[1], Ricardo de Almeida Falbo[2], Maria-Eugenia Iacob[3], Marten van Sinderen[3], Luís Ferreira Pires[3]
[1]Federal Institute of Espírito Santo, Colatina, Brazil, [2]Federal University of Espírito Santo, Vitória, Brazil
[3]University of Twente, Enschede, The Netherlands

Abstract— **ArchiMate is a widely adopted enterprise architecture modeling language that includes the "service" construct as a key structuring element across its enterprise layers. A previous analysis of the use of this construct within ArchiMate's business layer concluded that it fails to represent some important social aspects associated with the dynamics of service relations, which led to recommendations for improvements in the form of modeling patterns with focus on the business layer of ArchiMate. In this paper, we extend that analysis to consider also service relations in the application and technology layers. We explore the importance of addressing two complementary views for service modeling: the capability-based and the commitment-based views. As a result, a more comprehensive modeling strategy for service relations in ArchiMate is proposed; this strategy is able to reflect business models that employ the service notion, including software-as-a-service (SaaS), platform-as-a-service (PaaS), and infrastructure-as-a-service (IaaS). We use a reference ontology for services (UFO-S) to support our analysis.**

Keywords—ArchiMate; Service-Oriented Enterprise Architecture; service modeling; service commitments

I. INTRODUCTION

The service notion has had significant impact in enterprise architecture practices defining what has been called Service-oriented Enterprise Architecture (SoEA) [1]. In a typical SoEA, services act as a means to structure architectural elements within an architectural layer (e.g., in a network of enterprises related through business services) as well as a means to link different layers, with "higher" architectural layers accessing the resources of the "lower" layers by means of services (e.g., with IT services supporting business services) [2][3].

A prominent example of a SoEA approach is ArchiMate (currently a specification maintained by The Open Group) [4]. It includes a conceptual framework and also a modeling language for SoEA description [2] with the service construct as a basic primitive. In ArchiMate, a service is defined as "a unit of functionality that a system exposes to its environment" and is used throughout its three layers, giving rise to business, application and infrastructure services.

An analysis of the use of the service construct within ArchiMate's business layer concluded that it fails to represent some important social aspects associated with the dynamics of service relations [5]. These problems were revealed by employing a core reference ontology for services called UFO-S [6], which is based on the key observation that services are provided/consumed in a network of social relationships that evolves throughout the service lifecycle, encompassing service offering, service negotiation and service delivery. Each phase in the service life-cycle leads to different kinds of *social commitments and claims* established between service participants [7], which ultimately drive service delivery when the necessary resources/capabilities are employed to fulfill the commitments established in the agreement. The analysis resulted in three modeling patterns to represent unambiguously the notions of "service offering type", "service offering" and "service agreement" [5] in ArchiMate's business layer.

In this paper, we expand the scope of investigation considered in [5] and address the service construct beyond the business layer. We consider service relations in the application and technology layers, as well as the so-called cross-layer dependencies, which represent the relation between the business layer and the application and technology layers. We explore the importance of two complementary service modeling views: the capability-based and the commitment-based views. The former is prevailing in ArchiMate, whereas the latter is relatively neglected in the language [5][8].

We aim at defining a modeling strategy for service relations in ArchiMate (encompassing general guidelines and modeling patterns) that is able to support each of the complementary views, exposing also their relation in combined views when needed. As a result, the modeling strategy we propose enables modelers to represent software-as-a-service (SaaS), platform-as-a-service (PaaS), and infrastructure-as-a-service (IaaS) scenarios [9]. We use the existing constructs of *product* and *contract* to reveal the commitment-based perspective. These constructs are currently used in ArchiMate only at the business layer; here we propose they should be considered core constructs, and, as such, could also be used to represent aspects of application and infrastructure services.

This paper is further organized as follows: Section II presents the conceptual foundations for services used in this paper, in particular it discusses how UFO-S [6] conceptualizes service relations and how it harmonizes the capability-based and the commitment-based perspectives on services; Section III presents an overview of service modeling in ArchiMate, as well as previous work addressing service relations at the business layer; Section IV reveals limitations in ArchiMate with respect to service modeling in the light of UFO-S and the commitment-based perspective; Section V presents modeling recommendations for the identified limitations; Sections VI and VII present related work and final considerations.

978-1-4673-9933-3/16/$31.00 ©2016 IEEE

II. CONCEPTUAL FOUNDATION FOR SERVICES

The complex and multifaceted notion of service has led to a number of service characterizations, among which there are those that directly associate service provisioning with the use or application of resources and capabilities [10]. This *capability-based view* is discussed under different banners, including: "service as capability" (capability of a provider to produce benefits to customers) [11][12]; "service as application of competences" (manifestation of one party's capability to act for the benefit of another party) [13], and "service as resource/capability applied in process" (an integrated view between service as resource and service as process) [14].

Other characterizations of service appeal to the notion of "service as commitment" [7][6]. A number of works in Service Science [15][7][16] and Service Computing [17][18][19] explicitly mention commitments, promises and/or obligations for characterizing the service relations established between service participants. The benefits of a characterization based on commitments have been discussed from the perspective of business [15][16] as well as IT [20]. In the context of Service-Oriented Architectures (SOA), Singh and colleagues [20] remark that commitments can be used for raising the low-level abstraction of existing SOAs, allowing to reduce the gap between the business and the IT perspectives. In their view, commitments capture business meaning, which is not directly represented in process-oriented approaches [20], since process-oriented approaches focus on the sequence of tasks in which resources/capabilities are used/applied.

We consider both perspectives to be complementary and required in a thorough account of services, especially if this account is to be used as a foundation for SoEA. This has been one of the motivations in the development of a core reference ontology for services called UFO-S [6]. UFO-S aims at harmonizing different service perspectives, such as "service as commitment", "service as capability", but also "service as behavior", and "computational services" [6].

In UFO-S, service relations are fundamentally characterized by the commitments (and reciprocal claims) established between service agents (customers, providers) [6], which drive the dynamics of the service life-cycle. UFO-S focuses on the three basic phases of the service life-cycle [6]: (i) service offer (when a service offering is made available to a target community), (ii) service negotiation (when providers and customers negotiate to establish a service agreement), and (iii) service delivery (when actions are performed to fulfill a service agreement by employing capabilities).

A service offering is an entity that mediates the social relations between the service provider and the target customer community. A service offering is composed of service offering commitments from the service provider towards the target customer community, and the corresponding service offering claims from the target community towards the service provider. The content of the service offering commitments and claims may be described in service offering descriptions (e.g., folders, registration documents in a chamber of commerce, and artifacts in software service registries) and may refer to resources/capabilities to be used in the service provisioning.

Service provider is the role played by agents (e.g., physical agents such as persons, and social agents such as organizations [21]) when they commit themselves to a target customer community through offering commitments. A target customer community is the collective formed by the agents to which the service is offered. Target customer is the role played by agents when they become members of the target customer community, and, as a consequence, have claims for the fulfillment of the commitments established by the service provider agent.

Once a service is offered, service negotiation may occur. If service negotiation succeeds, a service agreement is established, and the service provider starts to play the role of hired service provider, while the target customer starts to play the role of service customer. A service agreement mediates the social relations between service customer and hired service provider, being composed of commitments and claims. Service agreements involve not only commitments from the hired service provider towards the service customer, but may also involve commitments from the service customer towards the hired service provider (e.g., the commitment to pay). Hired provider commitments and claims are properties that inhere in a hired service provider and are externally dependent on a service customer. Service customer commitments and claims are properties that inhere in a service customer and are externally dependent on a hired service provider. The content of commitments/claims of a service agreement may be described in a service agreement description (e.g., contracts) and may make reference to the resources and capabilities to be used in a particular service provisioning.

The mutual service commitments/claims established in the service agreement drive service delivery. The service delivery phase concerns the execution of actions, use of resources, and application of capabilities (from provider and customer) necessary to fulfill the service commitments mutually established in the service agreement.

Note that possessing resources/capabilities to perform certain actions or to produce certain outcome is not sufficient for characterizing a genuine service relation [6]. For example, the capability of an organization to wash cars is not sufficient to make it a car wash service provider. In fact, even an organization that is not capable of washing cars may offer a car wash service, delegating the actual washing to a capable third party. Thus, the picture is only complete by considering capabilities and the commitments that influence the application of capabilities [6] [8].

Finally, an important aspect of this theoretical foundation is that service relations are inevitably a social phenomenon between intentional agents [8]. Only intentional agents play the roles of service provider and service customer, since only this kind of agent can establish commitments to other agents. As a result, enterprise resources such as application components and infrastructure nodes do not themselves play the role of service providers and customers. Instead, service provider and service customers (agents) employ resources (such as application components and infrastructure nodes) as a means to fulfill their commitments.

III. SERVICE MODELING IN ARCHIMATE

In this section, we present an overview of the service modeling elements according to ArchiMate's metamodel, and describe a previous effort [5] that analyzed service modeling at the business layer in terms of UFO-S.

A. An Overview of ArchiMate Modeling Elements

ArchiMate defines a layered structure by means of which the enterprise elements can be organized. According to this structure, the lower layers support the higher ones through service provisioning [4]. There are three layers in ArchiMate: the business layer, the application layer, and the technology layer. The *business layer* deals with, among others things, business processes, people and organizations offering/hiring products/services [4]. The *application layer* supports the business layer with application services realized by software applications [4]. The *technology layer* supports the higher layers by providing infrastructure services (e.g., storage and communication services) realized by software and hardware systems (e.g., network devices, and DBMS) [4].

Fig. 1 presents a fragment of the ArchiMate metamodel focusing on the business layer elements used in this paper. A *business service* is "a service that fulfills a business need for a customer (internal or external to the organization)", and may be assigned to business interfaces. A *business interface* is a "point of access where a business service is made available to the environment" (e.g., phone, website, etc.), and it may be modeled as part of a business role. A *business actor* is "an organizational entity that is capable of performing behavior" (e.g., a person, an organization). A *business role* is "the responsibility for performing specific behavior, to which an actor can be assigned" (e.g., Attendant, Cable TV Company). Business actors and business roles can use business services through business interfaces [4]. A *product* is "a coherent collection of *services*, accompanied by a contract/set of agreements, which is offered as a whole to (internal or external) customers" [4]. A *contract*, in turn, is "a formal or informal specification of agreement that specifies the rights and obligations associated with a product" [4]. To illustrate the usage of the aforementioned elements, we can consider a case in the Cable TV domain in which "Easy TV, Inc." is a business actor playing the role of a "Cable TV Company" offering the "Cable TV" and the "Customer Support" business services to customers as part of a "Special Cable TV" product, which is driven by terms and conditions described in a contract.

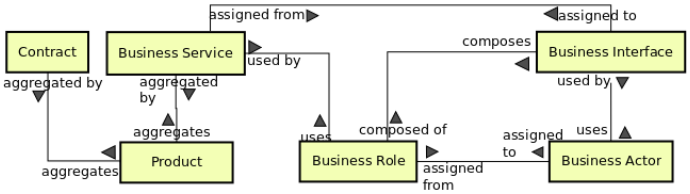

Fig. 1. A Business layer metamodel fragment.

Regarding the application layer, Fig. 2 presents a fragment of the ArchiMate metamodel with the elements used in this paper. An *application service* is "a service that exposes an automated behavior". An *application component* is "a modular, deployable, and replaceable part of a software system that encapsulates its behavior and data, and exposes these through a set of interfaces". Thus, application components realize application services in the sense that they execute the functionality exposed as a service. Application components can also use application services through a kind of *application interface*, so-called in ArchiMate "application-to-application" interfaces (e.g., an API) [4]. An *application interface*, in turn, is "a point of access where an application service is made available to a user or another software application". In the hypothetical Cable TV scenario, the "Easy TV, Inc." could count on a "Complaint Management System" as an application component that makes available a "Record Complaint" application service accessible through a "website" interface.

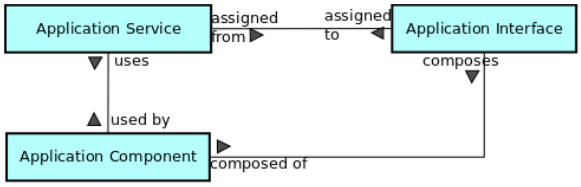

Fig. 2. An Application layer metamodel fragment.

Fig. 3 presents a fragment of the ArchiMate metamodel with the technology layer elements used in this paper. An *infrastructure service* is "[…] an externally visible unit of functionality, provided by one or more nodes, exposed through well-defined interfaces, and meaningful to the environment". An *infrastructure interface* is "a point of access where infrastructure services offered by a node can be accessed by other nodes and application components". Finally, *node* is "a computational resource upon which artifacts may be stored or deployed for execution". Nodes realize infrastructure services. In our scenario, the "Easy TV, Inc." has a "Database Server" node that, through the "ODBC" interface, makes available the "Data access" infrastructure service.

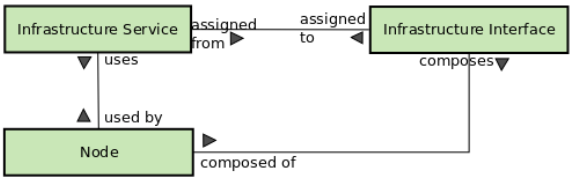

Fig. 3. A Technology layer metamodel fragment.

In addition to the relationships between elements of the same layer, ArchiMate defines cross-layer relationships. Fig. 4 presents an ArchiMate's metamodel fragment with focus on these relationships. Note that, as a "collection of services", a *product*, even though located in the business layer, can "aggregate" application and infrastructure services in addition to business services. *Application services* can be "used by" business actors and business roles directly. In this case, the service functionality is available through an "application-to-business" [4] interface (e.g., a website) by means of which business role/actor can use the service. Further, a *business service* may be "assigned to" an *application interface*, in which case this service is said to be "fully automated" by an application component that has this interface. Finally, the technology layer supports the application layer through *infrastructure services*, which can be "used by" application components. In the hypothetical Cable TV scenario, the "Database Server" node, by means of an "ODBC" infrastructure interface makes available the "Data access"

infrastructure service, which is used by the "Complaint Management System" at the application layer. This system, in turn, makes available the "Register Complaint" application service by means of the "website" as an 'application-to-business' interface. This application service is then used for supporting the provisioning of the "Customer Support" business service available to the Customer.

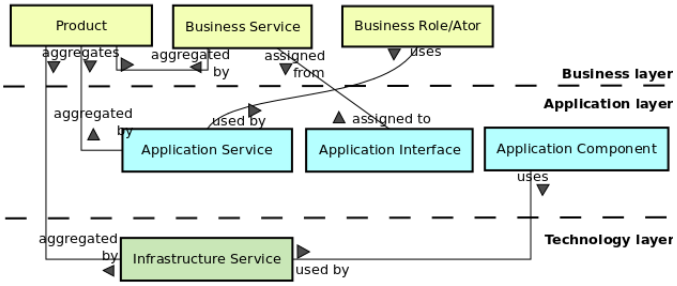

Fig. 4. A metamodel fragment with focus on some cross-layer dependencies.

Table I presents the notation of the aforementioned ArchiMate's modeling elements and relationships.

TABLE I. ARCHIMATE'S MODELING ELEMENT/RELATIONSHIP NOTATIONS.

Elements			
Name	**Notation**	**Name**	**Notation**
Business actor	Business Actor	Application Component	Application Component
Business role	Business Role	Application Interface	
Business Interface		Application Service	Application Service
Business Service	Business Service	Infrastructure Interface	
Product	Product	Node	Node
Contract	Contract	Infrastructure Service	Infrastructure Service
Relationships			
Name	**Notation**	**Name**	**Notation**
Used by		Realization	
Assignment		Composition	
Aggregation			

B. *Previous Work on Service Modeling at the Business Layer*

In [5], an analysis of service modeling at ArchiMate's business layer using UFO-S was reported. From this analysis, it was possible to identify service modeling limitations in ArchiMate to distinguish service offering types, service offerings and service agreements. For example, consider the model fragment of Fig. 5, which presents two different cable TV companies ("Easy TV, Inc." and "Smart TV, Inc.") offering two services ("Cable TV" and "Customer Support") that take part in a product ("Special Cable TV"). The terms and conditions of the product are described in a contract, and "John" and "Mary", as "Cable TV Customer", use the services. However, considering this model fragment, we cannot answer questions such as: (i) Are "John" and "Mary" target customers or are they actual service customers (i.e., have they established a service agreement with a hired service provider)? (ii) Among which service participants ("John", "Mary", "Easy TV, Inc." and "Smart TV, Inc") is the "Special Cable TV Contract" established? (iii) Does "Special Cable TV Contract" represent a specific service agreement between a hired service provider and a particular customer, or is it a type of service contract that applies to any target customer?

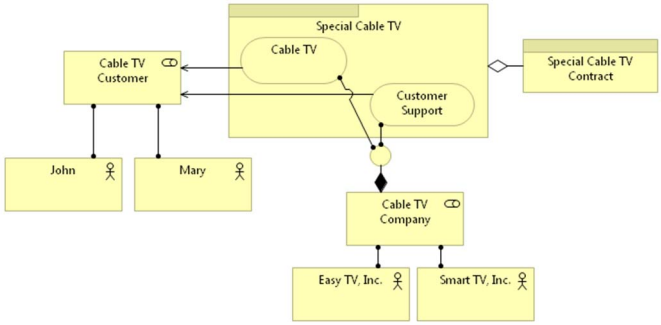

Fig. 5. Motivating example: a Cable TV scenario.

In order to address these and other limitations, the authors offered recommendations in the form of three modeling patterns: a service offering type pattern, a service offering pattern, a service agreement pattern. These modeling patterns were given real-world semantics based on UFO-S. The proposed modeling patterns use the existing *service, product* and *contract* modeling elements, as well as the *association* relationship [5]. Each pattern is composed by four groups of elements: (i) a *product* and related *services*, (ii) the *roles/actors* that provide the product/service, (iii) the *roles/actors* that consume the product/service, and (iv) the respective *contracts*. The *contracts* are in the center of each modeling pattern, representing service commitments and claims as fundamental elements in service relations. The associations in which a contract is involved establish the semantics of each pattern: in a service offering type, the contract connects the provider role with the service customer role; in a service offering, the contract connects the service provider actor with the customer role; and (iii) in a service agreement, the contract connects the (hired) provider actor with a particular customer actor.

Fig. 6 presents an example with the service offering pattern.

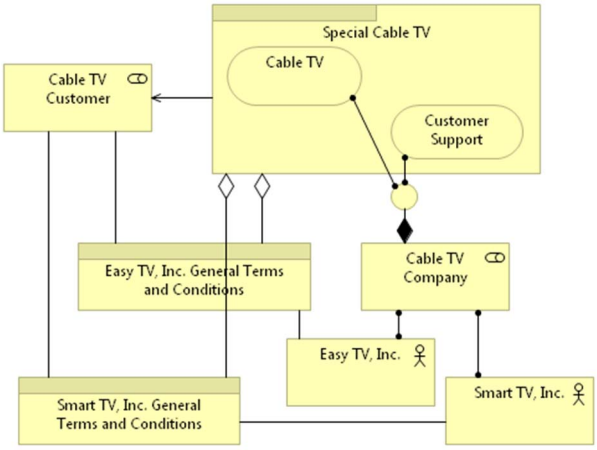

Fig. 6. The service offering pattern applied twice [5].

There are two service offerings: one from "Easy TV, Inc.", and another from "Smart TV, Inc," (both as service providers). The offerings are for (target customer) individuals that can (in case of an agreement) act as "Cable TV Customer" (i.e., service customers). The "Easy, Inc. General Terms and Conditions", and the "Smart TV, Inc. General Terms and Conditions" contracts, in turn, associates the respective service providers to the service customer role.

Fig. 7 presents an example with the service agreement pattern. The model illustrates a service agreement between "Mary" and "Easy TV, Inc.", which play, in this service agreement, the roles of service customer and hired service provider, respectively. Terms and conditions of this agreement are described in the "Mary-Easy TV, Inc. Contract", which associates the two individuals involved in the agreement.

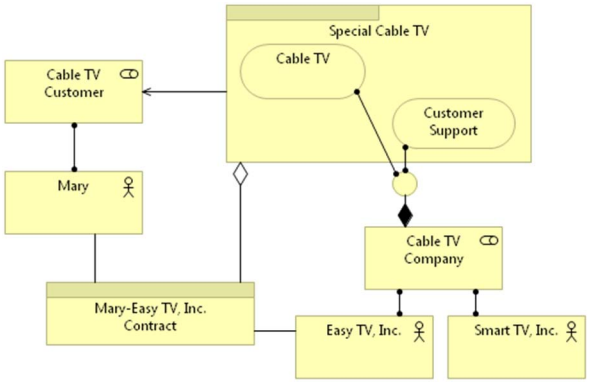

Fig. 7. An example of the service agreement modeling pattern [5].

These modeling patterns use the product and contract elements to complement the capability-based view and thereby reveal the commitments established by service participants at the business layer. In the following sections, we extend the use of these patterns to address application and infrastructure services, motivated by the limitations discussed in Section IV.

IV. ARCHIMATE SERVICE MODELING LIMITATIONS

We illustrate the limitations of the capability-based view with the well-known "Dropbox File Hosting" service operated by "Dropbox, Inc.". We select a modeling element to represent this service (in this case either an application service or a business service) and discuss the implications of the available choices. We explore in the model fragments of those modeling elements from the ArchiMate metamodel that can be related with the service element at hand (application components, application interfaces, business roles, business interfaces, etc.). We use UFO-S concepts, underlining them for emphasis.

Let us first assume that "Dropbox File Hosting" should be modeled as an application service. This choice is reflected in Fig. 8. Note that the service is accessed through an 'application-to-business' [4] interface (in this case a "website") by a "File Hosting Customer" (service customer) who uses the service for uploading, downloading and sharing files. The service is realized internally by the "Dropbox Software Application" component. This model represents the service customer role explicitly. While it represents the technical resources applied to realize the service, it fails to reveal who plays the role of service provider role ("Dropbox, Inc.").

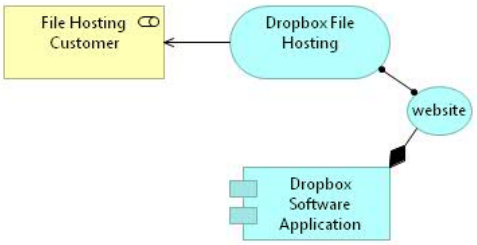

Fig. 8. Dropbox File Hosting as an application service used by business role.

Alternatively, we could also model the "Dropbox File Hosting" service as an application service used directly by other applications (e.g, the "Microsoft Office") for sharing files and information. In this case, the applications would access the service through an 'application-to-application' [4] interface, e.g., an Application Programming Interface (API), as Fig. 9 shows, in a pure intra-layer model. Note that, with this model, we can not reveal the service provider and the service customer (roles) involved in the service relation, focusing solely on the non-intentional resources (application components) that contribute to the service delivery.

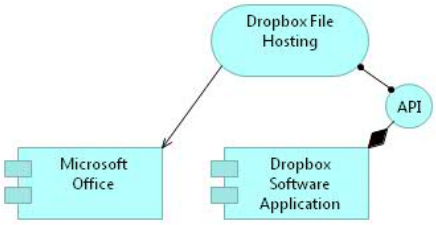

Fig. 9. Dropbox File Hosting as application service used by an application.

These limitations in the representation of application services seem to have roots in the definition of application service in ArchiMate. An application service is defined as "[…] a service that exposes automated behavior", i.e., a "unit of functionality" realized by a technical resource, here, an application component (with its capabilities). Clearly, this is only part of the broader business reality in this case: by having established a service relation to "Dropbox, Inc.", "Mary" (and other service customers) can use the functionalities of the "Dropbox Application Component" made available by means of the "Dropbox File Hosting" application service. These aspects remain underexplored in ArchiMate when the service is modeled as an application service.

Let us represent "Dropbox File Hosting" as a business service instead. In this case, there are two alternatives: (i) to consider it as an automated business service, if assigned to an application interface or (ii) to consider it as a regular (non-automated) business service, if assigned to a business interface. As an automated business service, it would be assigned to the application interface related to the application component that automatically performs the service, as Fig. 10 shows. This model fragment tries to reveal the business perspective associated with the service relation by using the business service element. However, despite that, it is still unable to represent important aspects of this business reality. Here, similarly to the case in which it is considered an application service, the service is assigned to the application interface,

preventing us from revealing the intentional agent that acts as the service provider ("Dropbox, Inc."). Thus, even if we try to represent "Dropbox File Hosting" as a fully automated business service, we cannot properly say who is the service provider committed to delivery it.

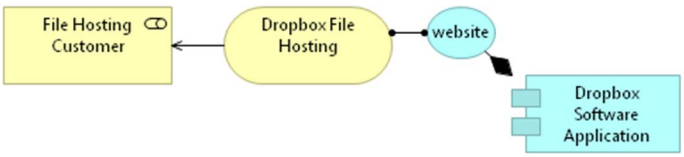

Fig. 10. Dropbox File Hosting as a fully automated business service.

If regarded as a non-automated business service, "Dropbox File Hosting" could be associated with "Dropbox, Inc." (service provider) through assignment to business interfaces used for service delivery ("Internet"), as Fig. 11 shows.

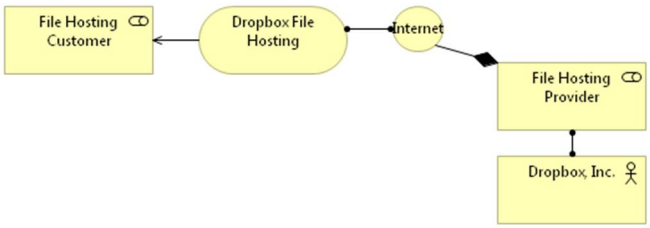

Fig. 11. Dropbox File Hosting as a non-automated business service.

In this case, however, it is unclear that "Dropbox, Inc." offers the service mainly by using technical resources/capabilities (e.g., application components), failing to characterize the SaaS business model that is inherent in this architecture. The model fails to reveal the relation between the service and the supporting "Dropbox Software Application".

This issue becomes evident when we consider the model in Fig. 12, which augments the model shown in Fig. 11 by including the "Live Support" business service, which includes technical support through an online chat. In this case, "Dropbox File Hosting" and "Live Support" are modeled in the same way, thus the model becomes incapable of distinguishing a business service that makes intensive use of technical resources/capabilities (such as "Dropbox File Hosting") from a service that is people-intensive (such as "Live Support").

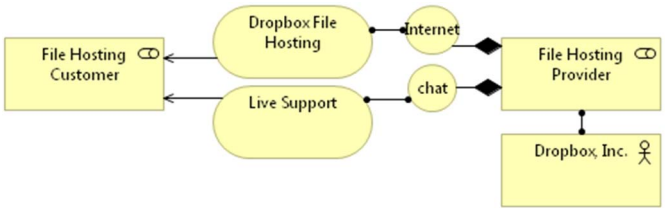

Fig. 12. Services with different characteristics, same modeling elements.

These limitations seem to have roots in the definition of business service in ArchiMate. A business service is defined as a "[…] service that fulfills a business need for a customer (internal or external to the organization)" [4]. The SaaS scenario exposes clearly that "fulfilling a business need" is not a characteristic that is restricted to business services. In fact, it is something that can also be achieved by using application and infrastructure capabilities. In the example, "Dropbox File Hosting" is provided by "Dropbox, Inc." to other companies or individuals and is realized by leveraging the "Dropbox Software Application". The same would apply to an infrastructure service: consider the "Amazon Simple Storage Service" provided by "Amazon, Inc." to "Dropbox, Inc." by leveraging the capabilities of the "Amazon Storage Server". Currently, application and infrastructure services lack this "marketing" perspective, as they are reduced conceptually to "units of functionality" fully realized by technical resources (application components and infrastructure nodes respectively).

In conclusion, the modeler would be forced to choose to represent "Dropbox File Hosting" either: (i) as an application service, (ii) as a fully-automated business service, or (iii) as a regular (non-automated) business service. In the first two cases, the business service provider/customer cannot be properly represented, failing to reveal an important commitment-related aspect; in the last case, while the service provider is represented explicitly, one fails to clarify that "Dropbox File Hosting" is actually a service that is provided mainly through application components (differently from a people-intensive business service), failing to reveal an important aspect of the capability-based perspective. Combining and reconciling both perspectives remains thus a challenge.

V. Modeling Recommendations

In this section, we address the identified limitations. We define some general guidelines (section V.A) and extended the modeling patterns initially proposed in [5] (section V.B).

A. General Guidelines

First of all, it is necessary to disentangle both perspectives in order to account for each of them adequately. We propose that, following the prevailing capability-based view in ArchiMate, regular *service* elements should be employed as usual, i.e., as behavior elements focusing on the "units of functionality" offered to the environment. The commitment-based view should be addressed by *products* and *contracts*.

In order to account for the capability-based view, it is necessary to rethink the concepts of business, application, and infrastructure services in ArchiMate. Since we observe that "fulfilling a business need" is not a particularity of business services, the criteria to classify services into business, application and infrastructure services should be related to whether the resources/capabilities used to provide the service are at the business, application or infrastructure layers (following the capability-based view). Thus, a service should be classified as a business service if this service is characterized as a "unit of functionality" that is performed by the manifestation of (a set of) capabilities inherent to intentional agents (e.g., person, departments, companies, etc.). Similarly, application and infrastructure services should be seen as "unit of functionality" performed by manifestation of capabilities inherent to, respectively, application components and infrastructure nodes.

In order to account for the commitment-based view, we propose that the product and contract elements should be considered core elements in ArchiMate (given them thus the same status as the "service" notion). If we consider their characterizations in ArchiMate (*product* is a "collection" /

"package" of different kinds of services offered to the environment, and contract as a means to specify "[…] rights and obligations associated with a product" [4]) these elements seem to be useful to represent commitment aspects in service relations. In terms of UFO-S, we understand that this bundle of services (and the underlying capabilities) from the service provider towards a service customer is driven by the commitments and claims established between them in a service relation. The notion of commitments/claims, therefore, is represented by means of the contract element that is directly related to the product. However, whereas resources/capabilities can be organized in an enterprise layered view, the service commitments established in service relations are not restricted to this structure and so the commitment-based view in ArchiMate should be considered as a crossed-layer view.

B. Extending the Service Modeling Patterns

As described in Section III.B, *products* and *contracts* are basic elements in the structure of the three proposed service modeling patterns. In the extended version of these patterns they continue to be such basic elements playing an important role for representing, respectively, the bundle of services and the service commitment/claims established in the service relations. However, the product element now encompasses services (and the corresponding capabilities) from different layers (not solely the business layer) and the contract represents the service commitment/claims that drive the application of capabilities independent of any enterprise layer. In the following, we present the usage of the extended modeling patterns, illustrating their applicability in harmonizing the capability-based view with the commitment-based view along all three ArchiMate enterprise layers.

Fig. 13 illustrates the "Dropbox File Hosting" service offering relation from "Dropbox, Inc." to its customers. The service relation between the service provider ("Dropbox, Inc.") and the customers is represented by means of product and contract elements, but without representing the services ("units of functionality"). Hence, the model explores solely the commitment-based view. The "File Hosting Customer" role is related to the product by means of a "used by" relationship, indicating the service customer role in the service relation. The service provider ("Dropbox, Inc.") is identified as the one who is related to the contract (using an "association" relation).

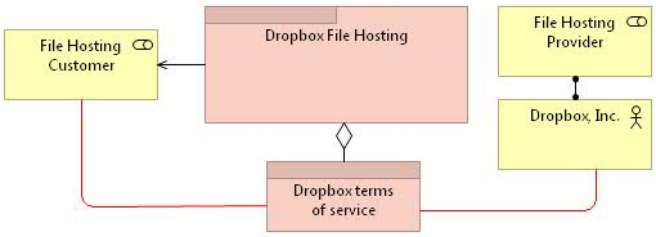

Fig. 13. Service offering relation from a commitment-based view.

Fig. 14 shows an example in which services that comprise the product are captured explicitly. These services represent the "units of functionality" made available to the service customer. We augment the scenario here and capabilities from business, application, and technology layers are made available to the customers in the form of business services ("Live Support"), application services ("Dropbox on the web"), and infrastructure services ("File Sync"). By means of the product and contract elements, we can represent that the service provider ("Dropbox, Inc.") is committed to guaranteeing the manifestation of these capabilities as described in the contract (describing the service offering commitments). In Fig. 14, therefore, the commitment-based view coexists with the capability-based view encompassing not only business service, but also application and infrastructure services.

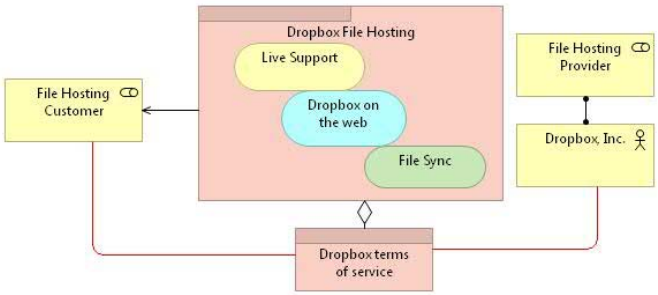

Fig. 14. Service offering relation: product and its services.

We can also augment the scenario by representing the resources (e.g., human resources, or technical resources) that make their capabilities available and can, therefore, realize (manifest) them during service delivery. Fig. 15 illustrates the resources used to realize the business, application and infrastructure services in the "Dropbox File Hosting" product ("Attendant", "Dropbox Software Application", and "Dropbox File Server"). Note that whoever or whatever interacts with the service is not necessarily the same individual that established the service relation. This is another important implication of the relation between the commitment-based and the capability-based views. While non-intentional objects, such as application components and devices, can interact with the service, and thereby "use" capabilities, they cannot be considered customers (in the sense of UFO-S) of the service relation. The "Live Support" business service and the "Dropbox on the web" application service are used by the "File Hosting Customer" (role). In this case, the "File Hosting Customer" represents the one who can hire these services as well as who can use them (in the sense of accessing the capabilities). The "Customer Computer" is a non-intentional object that can use the "File Sync" infrastructure service (i.e., accesses this capability) that was hired by the "File Hosting Customer".

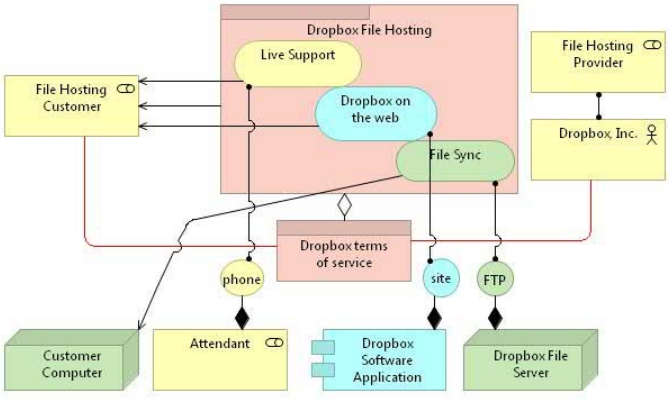

Fig. 15. Service offering relation and the used resources.

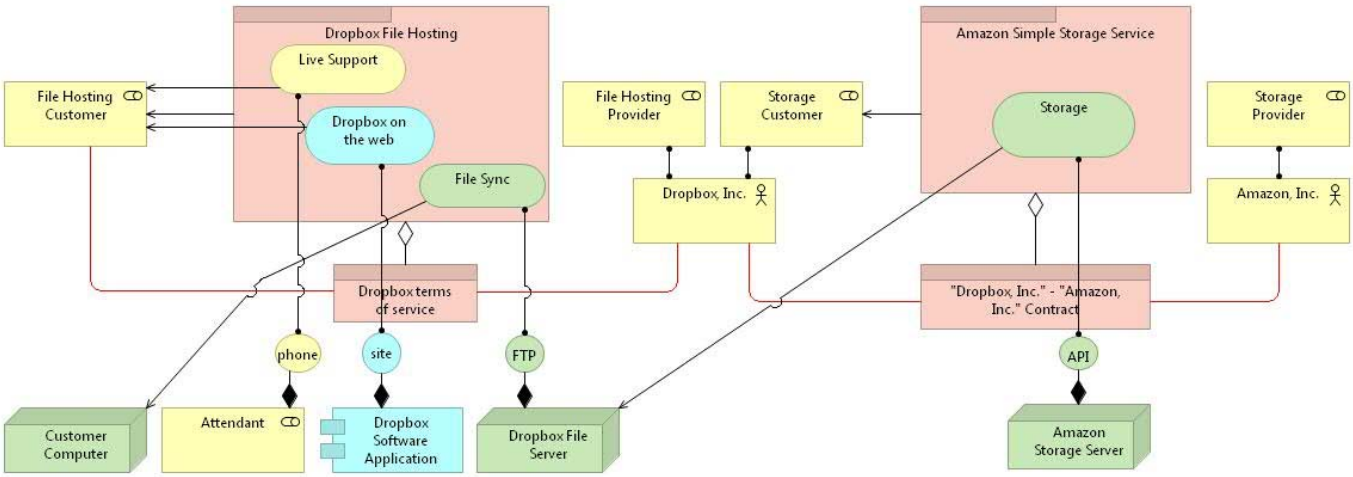

Fig. 16. Service offering and service agreement modeling patterns applied in tandem in an outsourcing scenario.

We should highlight that, although until here we have illustrated the extension of the modeling patterns structure by focusing on the service offering modeling pattern, the adaptations have their counterparts in the other proposed patterns. Fig. 16 illustrates the usage, in tandem, of two patterns (service offering and service agreement) inspired by a real outsourcing scenario that involved "Dropbox, Inc." and "Amazon, Inc." in the first eight years of Dropbox's operation [22]. In the left-hand side of Fig. 16, we can notice a service offering from "Dropbox, Inc." towards its customers. This offering encompasses the "Live Support" business service, the "Dropbox on the web" application service, and the "File Sync" infrastructure service. In the right-hand side of the figure, by applying the service agreement pattern, we represent that "Amazon, Inc.", as a hired service provider, is committed to make available the "Storage" infrastructure service for the service customer ("Dropbox, Inc."). By means of the service agreement established between these two intentional agents, "Dropbox File Server" device can use the service realized by the "Amazon Storage Server" node. Therefore, besides the commitment-based and capability-based complementary views, this model explores a service relation chain basically characterized by the service commitments established along the chain, which drive the application of the resources/capabilities. While in this paper we have illustrated service relations that cross enterprise boundaries, the approach is equally applicable to settings in which the services are offered by internal actors in the organization in a SaaS or IaaS strategy (consider, e.g., when Dropbox decided to insource their storage; in that case, "Amazon, Inc." would be replaced by an organizational unit of "Dropbox, Inc" playing the role of service provider).

Finally, in case of cloud service models, it is important that the application of the modeling patterns be accompanied by some representation strategies that reflect particular aspects of the cloud model, as well as the intended modeling view (capability-based or commitment-based view), as summarized in Table 2, and Table 3. Table 2 presents the representation strategies for each considered cloud service model. The capability-based representation reflects the kinds of resources employed predominantly in each service model (i.e., application components, system software or devices). The commitment-based representation is equally applicable across cloud service models, and reflects the service lifecycle dynamics (whether one is representing a service offering type, a service offering, or a service agreement). Table 3 presents the result of the combination of capability-based and commitment-based views, revealing the kinds of services that play a key part in the product element. The use of the commitment-based view opens us to possibility to bundle services of different kinds, which is quite common in the various cloud service models especially with ancillary services (e.g., a "telephone support" business service, an "administration panel" application service to manage virtualized devices).

TABLE 2. REPRESENTATION STRATEGIES FOR EACH CLOUD SERVICE MODEL.

Cloud service model	Capability-based representation	Commitment-based representation
Software-as-a-Service (SaaS)	(Predominance of) *application components* as resources to realize application services	Product and contract relating either: (i) service provider role and service customer role (service offering type) (ii) service provider actor and service customer role (service offering) (iii) service provider actor and service customer actor (service agreement)
Platform-as-a-Service (PaaS)	(Predominance of) *system software* as resources to realize infrastructure services	
Infrastructure-as-a-Service (IaaS)	(Predominance of) *devices* as resources to realize infrastructure services	

TABLE 3. COMBINATION OF CAPABILITY-BASED AND COMMITMENT-BASED VIEWS.

Cloud service model	Combined representation
Software-as-a-Service (SaaS)	*Application services* used as a key element in a *product*; other kinds of services may also be included in the product, e.g., ancillary business services.
Platform-as-a-Service (PaaS)	*Infrastructure services* (exposing the functionality of *system software*) used as a key element in a *product*; other kinds of services may also be included in the product, e.g., ancillary business and application services
Infrastructure-as-a-Service (IaaS)	*Infrastructure services* (exposing the functionality of *devices*) used as a key element in a *product*; other kinds of services may also be included in the product, e.g., ancillary business and application services

VI. Implications to Service Representation Approaches

In this section, we discuss how the service modeling strategy proposed in this paper is related to, and/or can benefit three service representation approaches that, despite their particularities, try to address service provisioning considering technical and business/market aspects.

SoaML (Service oriented architecture Modeling Language) is a UML (Unified Modeling Language) profile and a metamodel by the OMG (Object Management Group) for designing services in service-oriented architectures [23]. In SoaML, a service is "[…] a resource that enables access to one or more capabilities" [23]. Also, "providers" and "consumers" of service may be people, organizations, technology components or systems [23]. A service contract, in turn, is a binding on both the providers and consumers and separates concerns of how parties agree to provide and uses the service from how the parties implements their role [23]. SoaML, however, does not offer a clear distinction between intentional and non-intentional providers/consumers. As a consequence, a clear distinction between capability and commitment-based aspects is somehow compromised. We believe therefore that SoaML could first establish a clear distinction between intentional agents that establish genuine service relations and non-intentional resources applied to realize the services. Thus, the language could benefits in the separation of concerns proposed in this paper regarding the capability-based and commitment-based views. It could also improve the representation of service provisioning by addressing, e.g., business/market partners from one side, and how their resources can interact (through, e.g., interfaces, outputs, inputs, conditions) from another.

USDL (Unified Service Description Language) [24] is a platform-neutral language for service description supported by SAP Research that aims to unify three aspects, namely: business (involving, service provider and other roles), operational aspects (e.g., capabilities, dependences, compositions), and technical aspects (e.g., protocols, and technical interfaces). The language's elements are organized in modules, among which we point out three of them: *participants*, *service*, and *service level*. The *participants* module is related to the actors that participate in the provisioning, delivery and consumption of services (e.g., business owner, provider, intermediary, and target consumer) [24]. The *service* module captures central service concepts, e.g., *service* and *service bundle* [24]. The *service level* module captures concepts concerned with guarantees of quality of service, which are claimed/requested by different actors involved with the service [24]. USDL, by means of the "service bundle" concept, allows to combine services from a people-intensive perspective (e.g., "freight service"), as well as from a technical one (e.g., "web services") in a whole service provisioning (e.g., "carrier service"). Despite that, however, it is not clear in USDL what characterizes these services from different perspectives (business/market from technical ones). As such, we believe that the aspects discussed in this paper regarding the characterization of different kinds of service (such as, business, application and infrastructure services) grouped as a bundle of capabilities in products, and whose provisioning and usage are driven by service commitments established between provider and customers involved in service relations, could enrich USDL's service representation.

Finally, SOMF (Service-Oriented Modeling Framework) is a modeling framework that focuses on software design [25]. It is not properly an enterprise architecture representation approach, but it offers a Cloud Computing Toolbox Model Language [25] that is related to some aspects discussed in this paper. In SOMF, the term "service" is referred as an "organizational software entity" (e.g., software system, web service, database, and business process) that encapsulates business requirements [25]. By being related to the capability-based view (focusing on describing how the resources are being employed as services), SOMF does not address explicitly some commitment-based aspects inherent to service relations. Thus, we believe that the commitment-based view advocated by us, could contribute for improving the understanding on how business requirements are encapsulated in the services. Usually, technical services are designed for fulfilling commitments established between two intentional agents (e.g., people, and departments). Thus, it would be possible to establish traceability links from the IT services to the service commitments that motivated their design. Also, we believe that the notion of service description (and the underlying commitments/responsibilities) could be improved in SOMF especially for addressing aspects related to the representation of service offerings and service agreements relations.

VII. Final Considerations

In this paper, we proposed a service modeling strategy in ArchiMate able to reflect business models that employ a market as well as a technical notion of service. This contributes to harmonize and define means to represent the prevailing capability-based view in ArchiMate together with a commitment-based one (related to social/market aspects). This strategy is then useful for addressing currently widely adopted cloud service models, such as, SaaS, PaaS, and IaaS.

For that, we have conducted an ontological analysis in the light of a service ontology (UFO-S), by means of which we could identify some service modeling limitations in ArchiMate for properly representing service relations at business, application and technology layers. In order to address these limitations, we reconsider some aspects in ArchiMate service modeling in a form of modeling guidelines, which were incorporated in an extension of the service modeling patterns initially proposed in [5]. By using the modeling patterns, the modelers are able to represent service relations (service offerings, and service agreements) not only at the business layer, but also at the application and technology layers.

The proposed solution as a whole characterize a kind of lightweight solution that avoids profound modifications of ArchiMate's metamodel, and intends to favor easy adoption by modelers for obtaining benefits of expressiveness and clarity directly. Despite that, however, we understand that some aspects should be inevitably reconsidered in ArchiMate specification. We advocate then that the *product* and *contract* elements should not be restricted to the business layer, but be considered as ArchiMate core elements. This is justified by the fact that although enterprise resources/capabilities can be organized in an enterprise layered structure, the service

relations are cross-layered encompassing business, application and technology layers and the product and contract elements are useful for expressing this social/market view. Also, we suggest that the concept of "service" should be restricted to the fundamental notion of "unit of functionality" avoiding to incorporate a business/market view, which would be addressed by other more appropriate elements, such as product and contract. Thus, through the services we could address the capability-based view and when associated to product/contract we could incorporate the commitment-based view. Moreover, from our analysis we could indicate that some modeling choices possible in the current ArchiMate's specification are unnecessary, or not properly defined. For example, the notion of "fully automated" business service (as a people-intensive service) is unnecessary, since it can be replaced by a regular application service (used in the context of a particular service relation, which gives it a "business sense").

Finally, from this work we have as a future research agenda, the following: (i) to apply the proposed solution in the representation of real scenarios of cloud service models in order to evaluate it and explore the particularities of these scenarios regarding how resources/capabilities from different layers are made available by means of services in a market-based view; (ii) to clarify the semantics of the *product* element as a "bundle of services" in ArchiMate (which will also require a clarification of the semantics of *aggregation* and *composition* relationships between product and services), and (iii) to explore additional languages and/or formalisms to express the content of service commitments, i.e., the conditions, requirements, and constraints which are usually captured in service offering and service agreement descriptions and that ultimately drive service delivery. Moreover, we plan, as a natural extension of this work, to propose changes in the ArchiMate metamodel in order to incorporate the modeling aspects discussed in this work. We expect that the service modeling patterns and guidelines discussed here could lead to additional service modeling viewpoints in ArchiMate for representing cloud service models.

ACKNOWLEDGMENT

This research is partly funded by the Brazilian Research Funding Agencies CNPq (grant numbers 311313/2014-0, 485368/2013-7, 461777/2014-2) and FAPES (grant number 69382549).

REFERENCES

[1] C. Kistasamy, A. Van Der Merwe, and A. D. La Harpe, "The Relationship between Service Oriented Architecture and Enterprise Architecture," in *14th IEEE Intern. Enterprise Distributed Object Computing Conference Workshops (EDOCW)*, 2010, pp. 129–137.

[2] M. Lankhorst, *Enterprise Architecture at Work: Modelling, Communication, and Analysis*. Berlin: Springer, 2005.

[3] M. Cases, D. A. Bodner, and B. Mutnury, "Architecture of Service Organizations," in *Introduction to Service Engineering*, G. Salvendy and W. Karwowski, Eds. Hoboken: John Wiley & Sons, Inc., 2010, pp. 109–134.

[4] The Open Group, "Archimate 2.0 Specification," The Open Group, Berkshire, UK, 2012.

[5] J. C. Nardi, R. de A. Falbo, and J. P. A. Almeida, "An Ontological Analysis of Service Modeling at ArchiMate's Business Layer," in *18th IEEE International Enterprise Distributed Object Computing Conference (EDOC)2*, 2014, pp. 92–100.

[6] J. C. Nardi, R. de A. Falbo, J. P. A. Almeida, G. Guizzardi, L. F. Pires, M. Van Sinderen, N. Guarino, and C. M. Fonseca, "A Commitment-based Reference Ontology for Services," *Inf. Syst.*, vol. 51, 2015.

[7] R. Ferrario and N. Guarino, "Commitment-based Modeling of Service Systems," in *Third International Conference, IESS*, 2012, pp. 170–185.

[8] J. C. Nardi, R. de A. Falbo, and J. P. A. Almeida, "Revealing Service Commitments in Service-Oriented Enterprise Architecture," in *The Sixth Workshop on Service oriented Enterprise Architecture for Enterprise Engineering*, 2014, pp. 286–295.

[9] C. Weinhardt, B. Blau, and J. Stober, "Cloud Computing - A Classification, Business Models, and Research Directions," *Bus. Inf. Syst. Eng.*, vol. 5, pp. 391–399, 2009.

[10] C. L. B. Azevedo, M.-E. Iacob, J. P. A. Almeida, M. van Sinderen, L. F. Pires, and G. Guizzardi, "An Ontology-Based Well-Founded Proposal for Modeling Resources and Capabilities in ArchiMate," in *17th IEEE International EDOC Conference*, 2013.

[11] OASIS, "Reference Model for Service Oriented Architecture 1.0: OASIS Standard." OASIS, pp. 1–31, 2006.

[12] T. Ruokolainen, "A Model-Driven Approach to Service Ecosystem Engineering," University of Helsinki, 2013.

[13] S. L. Vargo and R. L. Lusch, "Evolving to a new dominant logic for marketing," *J. Mark.*, vol. 68, pp. 1–17, 2004.

[14] B. Anderson, M. Bergholtz, and P. Johannesson, "Resource, Process, and Use – Views on Service Modeling," 2012, pp. 23–33.

[15] S. Alter, "Service System Fundamentals: Work System, Value Chain, and Life Cycle," *IBM Syst. J.*, vol. 47.1, pp. 71–85, 2008.

[16] W. Mingming and H. Youbei, "Design and Implementation of Service Commitment," in *International Conference on Service Sciences ICSS*, 2010, pp. 388–391.

[17] L. O. B. D. S. Santos, G. Guizzardi, R. S. S. Guizzardi, E. G. Da Silva, L. F. Pires, and M. Van Sinderen, "GSO: Designing a Well-Founded Service Ontology to Support Dynamic Service Discovery and Composition," in *13th Enterprise Distributed Object Computing Conference Workshops*, 2009, pp. 35–44.

[18] J. O'Sullivan, "Towards a Precise Understanding of Service Properties," Queensland University of Technology, 2006.

[19] M. Dumas, J. O'Sullivan, M. Hervizadeh, D. Edmond, and A. H. M. Ter Hofstede, "Towards a Semantic Framework for Service Description," in *IFIP TC2WG26 Ninth Working Conference on Database Semantics Semantic Issues in ECommerce Systems*, 2001, pp. 277–291.

[20] M. P. Singh, A. K. Chopra, and N. Desai, "Commitment-Based Service-Oriented Architecture," *Computer (Long. Beach. Calif).*, vol. 42, no. 11, pp. 72–79, 2009.

[21] G. Guizzardi, R. A. Falbo, and R. S. S. Guizzardi, "Grounding software domain ontologies in the Unified Foundational Ontology (UFO): the case of the ODE software process ontology," in *Proceedings of the XI Iberoamerican Workshop on Requirements Engineering and Software Environments*, 2008, pp. 244–251.

[22] C. Metz, "The Epic Story of Dropbox's Exodus from the Amazon Cloud Empire," *Wired Business*, 2016. [Online]. Available: http://www.wired.com/2016/03/epic-story-dropboxs-exodus-amazon-cloud-empire/. [Accessed: 15-Apr-2016].

[23] OMG, "Service oriented architecture Modeling Language (SoaML) Specification," 2012.

[24] A. Barros, U. Kylau, and D. Oberle, "Unified Service Description Language (USDL)." 2011.

[25] Methodologies Corporation, "Service-Oriented Modeling Framework (SOMF): Cloud Computing Toolbox Model - Language Specifications (version 2.1)," 2001.