On the Support for the Assignment of Active Structure and Behavior in Enterprise Modeling Approaches

Rômulo Henrique Arpini, João Paulo A. Almeida
Ontology & Conceptual Modeling Research Group (NEMO)
Computer Science Department, Federal University of Espírito Santo (UFES)
Av. Fernando Ferrari, s/n, Vitória, ES, Brazil
rharpini@inf.ufes.br; jpalmeida@ieee.org

ABSTRACT

This paper aims to analyze and review the support of the different kinds of active structure assignment in enterprise modeling techniques and frameworks, including ArchiMate, DODAF, and ARIS. Since we believe that these frameworks will be used in the description of an Enterprise Architecture in tandem with the detailed description of business processes, we also discuss the support for active structure allocation in processes modeling techniques, including XPDL, UML Activity Diagrams and BPMN in our analysis. We briefly describe each of the approaches and discuss their expressiveness with respect to the relations between the active structure of an organization (described in terms of actors, roles, etc.) and the behavioral aspects of the organization (described in terms of processes, activities, functions, etc.).

Keywords

Enterprise Modeling, Business Process Modeling, Active Structure, survey.

1. INTRODUCTION

Several approaches to enterprise modeling manage the complexity of an organization by describing the organization from different perspectives focusing on: (i) organizational structure (with actors, roles and organizational units); (ii) organizational activities (structured into business processes, and more recently, services); (iii) information systems that support organizational activities, and (iv) technical infrastructure to support information systems.

The need to relate the various partial descriptions is addressed in virtually all enterprise modeling approaches and has been recognized in Zachman's early work in 1987 [24]: "each of the different descriptions has been prepared for a different reason, each stands alone, and each is different from the others, even though all the descriptions may pertain to the same object and therefore are inextricably related to one another."

This need has led to the development of *relations between architectural domains* in enterprise architecture and enterprise modeling approaches [6]. One of these domains, namely that of organizational behavior, has received significant attention in recent years in the context of business process modeling and management. Business process modeling addresses the way enterprises organize their work and resources showing how they contribute to fulfill the enterprise's strategies [16]. Another important domain, that of organizational structure is strongly inter-related with the process domain. While the process domain focuses on "how" the business process activities are structured and performed, the organizational structure domain focuses on "who" performs these activities, i.e., which kinds of entities in an organization are capable of performing work.

Given the strong connection between the two architectural domains, we argue that any comprehensive enterprise modeling technique should explicitly establish the relations between modeling elements to represent business activities, called here *behavioral elements,* and those used to represent the organizational actors involved in these activities, called here *active structure elements.* Although most of the techniques offer some support for establishing these relations, the levels of support and expressiveness they offer vary significantly.

Properly representing the assignment of active structure elements and behavioral elements at design time is important to allow the comprehensive analysis of an Enterprise Architecture, e.g., from the perspectives of accountability, authorization, and responsibility of organizational actors with respect to the activities they execute. The assignment of active structure and behavioral elements also supports business process enactment and later phases of process management, such as monitoring and evaluation, as observed in [8].

In this paper we analyze and review the support of the different kinds of active structure assignment in enterprise modeling techniques and frameworks, including ArchiMate, DODAF, and ARIS. Since we believe that these frameworks will be used in the description of an Enterprise Architecture in tandem with the detailed description of business processes, we also discuss the support for active structure allocation in processes modeling techniques. Instead of addressing an exhaustive list of business process and workflow modeling techniques, we have included here developments that we believe are representative of a large number of process techniques. First, we have included XPDL since it was conceived as an interchange format for a number of process-related products, including "execution engines, simulators, BPA modeling tools, Business Activity Monitoring and reporting tools" [15]. Second, we have addressed the support provided in UML 2.0 Activity Diagrams and BPMN 2.0 because of their wide acceptance to represent business processes.

The main contribution of this paper is to provide an overview of the support for active structure assignment in existing enterprise modeling approaches. Since we have observed that the various approaches capture complementary aspects of active structure assignment, we believe this overview can support us in the identification of gaps for future search efforts towards a comprehensive enterprise modeling approach.

2. ACTIVE STRUCTURE ASSIGNMENT

We begin by presenting the concepts that are used to model both active structure and behavior in each of the techniques considered and then discuss the level of expressiveness in the integration of active structure and behavior.

2.1 ArchiMate

ArchiMate is a modeling language that offers an integrated architectural approach that describes and visualizes the different architecture domains and their underlying relations and dependencies, aiming to offer an unambiguous specification and descriptions of enterprise architecture's components and specially their relationships with a consistent alignment [19]. The language is currently standardized by The Open Group in its version 1.0 and used to support architectural descriptions produced using TOGAF [18].

The language distinguishes three layers with different abstraction levels: (i) the business layer, which offers products and services to external customers, realized in the organization by business processes performed by business actors; (ii) the application layer, which supports the business layer with application services which are realized by software applications; and (iii) the technology layer, which offers infrastructural services for software applications. Each one of these layers includes modeling constructs to represent active structure elements, behavioral elements and passive structure elements, as shown in Figure 1.

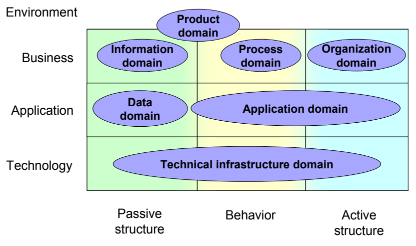

Figure 1 - Archimate Framework [19]

We focus on the concepts of the business layer, whose metamodel is presented in Figure 2. The abstract concept *Business Behavior Element* groups all concepts related to the behavioral structure. The link with the active structure is done through the assignment relationship, which allows a modeler to relate a *Business Behavior Element* to a *Business Role*. A *Business Role* may, in turn, be related to a *Business Actor* through an assignment relationship.

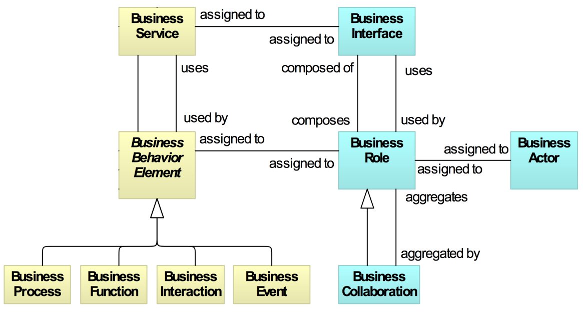

Figure 2 - Fragment of Business Layer Metamodel (from [19])

A *Business Actor* is an organizational entity capable of performing behavior, and performs the behavior assigned to one or more *Business Roles*. *Business Roles* are defined as a named specific behavior of a business actor in a particular context. A *Business Role* may be assigned to one or more business processes or *Business Functions*. *Business Processes* are defined as units of internal behavior or collections of causally-related units of internal behavior intended to produce products or services, while *Business Functions* are defined as units of internal behavior that group behavior according to some criteria, such as knowledge, resources and skills. A *Business Service* is an externally observable behavior that is realized internally by *Business Behaviour Element*s. A *Business Service* may be assigned to a role's *Business Interface*.

Figure 3 shows a small example of active structure assignment in ArchiMate, relating process, role and actor. The "ArchiSurance" actor is composed of two departments, namely, "Luggage Insurance Department" and "Travel Insurance Department". The "Travel Insurance Department" is assigned to the "Travel Insurance Seller" role, which is associated with the "Take out Insurance" process. Whichever actor is assigned to the "Travel insurance seller role" will perform the "Take out insurance process". In this specific example, the process should be performed by the "Travel Insurance Department". The example also reveals the assignment of the "Offering travel insurance" service (a behavioral element), by the means of a *Business Interface* provided by the "Travel insurance seller" role and realized by the "Take out insurance" process.

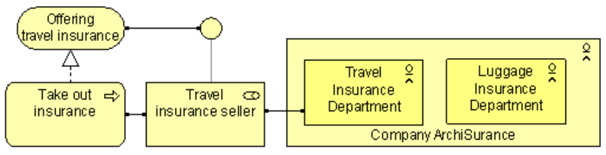
Offering travel insurance
Take out insurance
Travel insurance seller
Travel Insurance Department
Luggage Insurance Department
Company ArchiSurance

Figure 3 - Process, Actor and Role [19]

A *Business Collaboration* is defined as a temporary configuration of two or more roles, resulting in specific collective behavior in a particular context. Unlike the case in which a *Business Process* or *Function* is assigned to a single *Role*, *Business Collaborations* aggregates two or more *Roles*, meaning it represents a collective effort which may be more than the sum of the behavior of the separate roles. Collaborations are assigned to *Business Interactions*, which are used to describe the behavior that takes places within these collaborations. Figure 4 shows how both *Business Collaboration* and *Business Interaction* may be used. "Combined Insurance Seller" is the collaboration that aggregates the "Travel insurance seller" and "Luggage insurance seller" roles. The "Take out combined insurance" interaction involves the execution of the "Prepare travel policy" process, performed by the "Travel insurance seller" role, and the "Prepare luggage policy" process, performed by the "Luggage insurance seller" role.

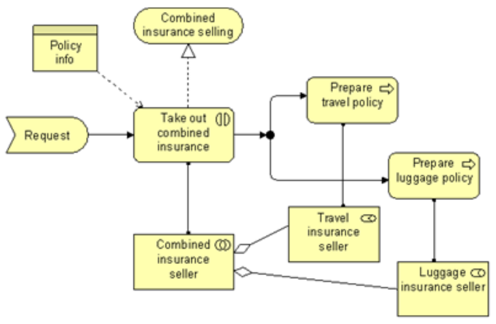

Figure 4 - Business Colaboration and Interactions [19]

2.2 DoDAF

The Department of Defense Architecture Framework (DoDAF) is a comprehensive framework and conceptual meta-model that has been designed specifically to meet the business and operational needs of the US Department of Defense [20]. Although the focus of the framework is clearly oriented to military systems, it can be extended to architectures that are more general [6], and provides concepts to model behavioral and active structure concepts.

In DoDAF, a *Performer* represents who may execute an *Activity*, and an activity represents specific operational actions. DoDAF introduces a few concepts to address the relation between performers and activities. A fragment of the metamodel with *Performer* and its related concepts is shown on Figure 5.

Performer is a type of *Resource*. A *Performer* may be further specified into one of the following types: (i) *System*, which is defined as "a functionally, physically, and/or behaviorally related group of regularly interacting or interdependent elements", (ii) *Service*, described as a "*Performer* to enable access to a set of one or more resources, such as Information, Data, Material and Performers"; (iii) *OrganizationType*, which is the type of an individual *Organization*. For example, we may have a "ForProfitOrganization" and "NonProfitOrganization" types; or (iv) *PersonType*, which defines a category of *IndividualPersons* that share common skills. A *PersonType* may also be used to represent a role that may be played with a more general *PersonType*, through the *personTypePartofPerformer* relation.

The linkage between an *Activity* and a *Performer* is given by the relationship *activityPerformedByPerformer*, which also has other associations: (i) *activityPerformableUnderCondition*, representing an activity that must be performed under certain conditions, e.g., "must be able to perform maneuver under Desert Conditions". It may also have a measure associated, quantifying the conditions under which the activity must be performed; (ii) *ruleConstrainsActivityPerformedByPerformer*, defining constraints under which a performer must perform an activity; (iii) *measureOfTypeActivityPerformedByPerformed*, which is a measure associated with the performance of an activity by a performer.

2.3 ARIS

The ARIS (Architecture for integrated Information Systems) method [14] is structured in four inter-related views (organizational, data, control, function) that support the description of an organization and its information system. The framework includes three abstraction layers (requirements, design and implementation), dealing with different levels of details separating specific concerns. The organizational view describes all the hierarchy of an organization, i.e., the communication and relationship between organization units and reveals the roles of the individuals in an organization, whereas the functional view is used to describe the tasks performed by the organization [14]. The control view shows the relationship between the business processes of an organization and the remaining entities of the organization (organizational structure, resources, information) of the business environment [3]. We focus on the control view at the requirements level.

Business processes in ARIS are modeled in Event-driven Process Chains (EPCs), whose main elements include *Functions*, *Events* and *Rules*. *Functions* are the main behavioral concept, representing organizational activities. *Functions* of an EPC can placed within swim-lanes, as shown in the example of Figure 6. In this example, a "Client" performs the "Request Purchase" *Function*, while a "Seller" performs the "Analyze purchase request" and the "Finish purchase" or the "Inform Client" *Functions*, depending on whether the purchase is approved.

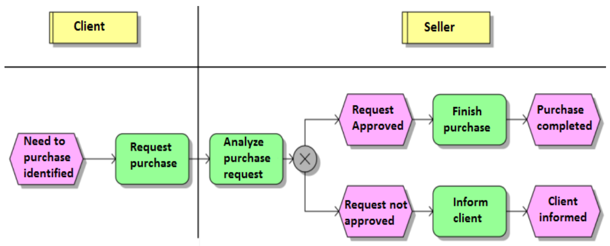

Figure 6 - Example of Business Process Model in ARIS [13]

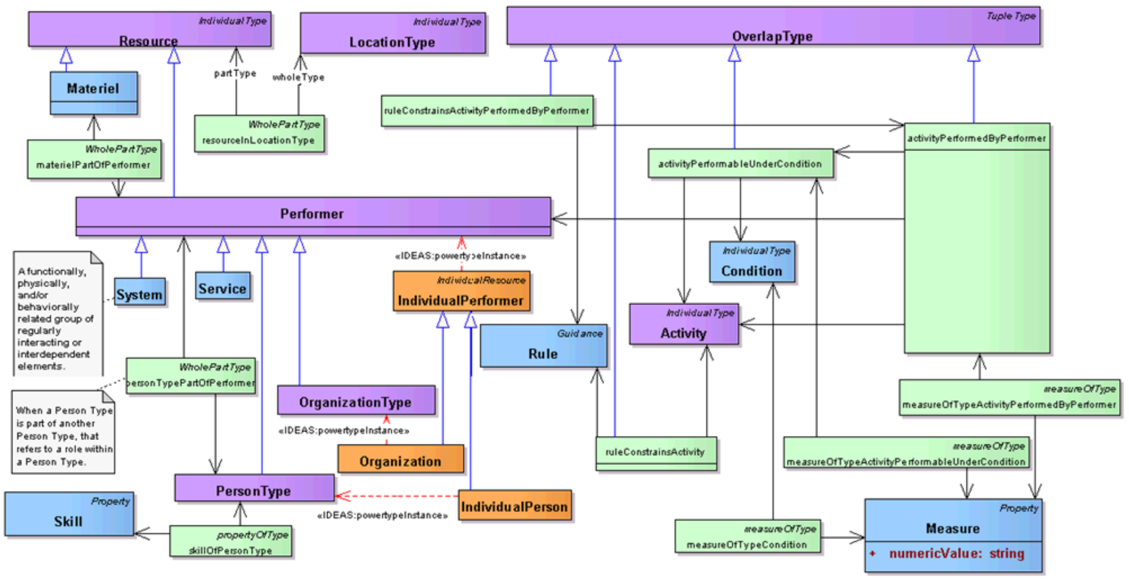

Figure 5 - Excerpt of DoDAF Performer Metamodel [20]

In our analysis, we consider the metamodel excavated in [12], where the authors have identified that the ARIS toolset recognizes the following relations between the active structure (represented by the abstract metaclass *Participant*) and behavioral (represented by *Function*): *is technically responsible for*, *carries out*, *is IT responsible for*, *decides on*, *must be informed about*, *contributes to*, *accepts*, *has consulting role in*, *must be informed on cancellation* and *must inform about result of*. *Carries out* is the main relationship, indicating who will be responsible for performing the *function*, and is the relationship represented in Figure 7.

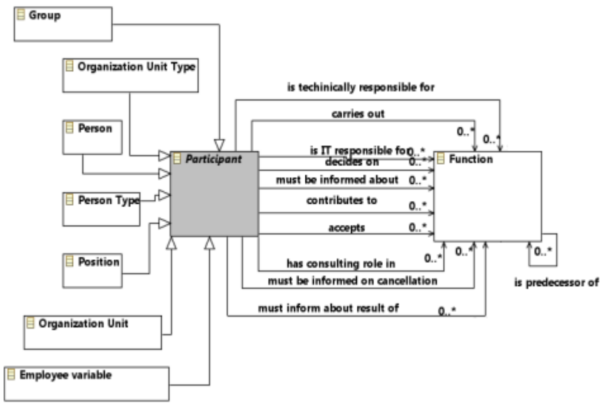

Figure 7 - Fragment of the Metamodel Adapted from [12]

Figure 7 also shows the organizational concepts used to describe the potential *participants* in the organizational activities, they are: *Organization Unit Type*, *Organization Unit*, *Position*, *Person Type*, *Person*, *Group* and *Employee Variable*. All of these concepts can be related with the *Function* concept through all the aforementioned relations. These elements are used in the so-called Organization Charts, which allows one to capture hierarchical and others active structure specific relations.

ARIS has a rich set of elements to describe organizational structure at instance-level and type-level. An *Organizational Unit* represents "an entity that is responsible for achieving organizational goals", being a real-world entity. An *Organizational Unit Type* is described as "a type of organization unit, i.e., an element that represents the common features (duties, responsibilities, etc.) of a set of organization units". A *Position* represents "the smallest organizational unit possible. The responsibilities and duties of a position are defined in the Position Description". A *Position Type* represents a "type of position, i.e. an element that represents the common features (duties, responsibilities, etc.) of a set of positions". A *Person* "is used to represent a person who is assigned to an organization". A *Person Type* represents a "generalization of person, i.e., an element that represents the common features (duties, responsibilities, feature, etc.) of a set of people". A Group represents "a group of employees (person) or a group of organizational units (Organizational Unit) that cooperate to achieve a goal". Finally, the semantics of the *EmployeeVariable* metaclass is not discussed in the ARIS documentation. ARIS also has a rich set of relations between those organizational structure elements, which include hierarchical relations (of technical and managerial nature), delegation relations, etc. We refrain from discussing them here due to space constraints (see [12] for a fuller description of the ARIS organizational metamodel).

2.4 XPDL

XPDL (XML process definition language) [22] was developed by the Workflow Management Coalition (WfMC) to support the interchange of workflow process definitions [21].

The topmost entity of an XPDL 2.1a model is a *Package*, which includes one or more process definitions [5] and one or more *Participant* definitions. A *Participant* represents the "description of resources that can act as the performer of the various activities in the process definition" [22]. Process definitions in a *Package* automatically "inherit" the *Participants* defined on that *Package*.

Figure 8 depicts the basic set of entities and relations for the exchange of process definitions. The entity *Participant* is further classified into one of the following basic types [22]: (i) *Resource*, when the participant represents a specific resource agent; (ii) *ResourceSet*, when the participant represents an aggregation of resources; (iii) *Organizational Unit*, when the participant represents is a department or any other unit within an organization model; (iv) *Human*, when the participant represents is a single person; (v) *System*, when the participant represents an automatic agent; (vi) *Role*, when the participant is a placeholder for a human which can perform a specific function. Note that XPDL does not provide a clear semantics for each one of the basic types.

Figure 8 shows an association between the *Participant* entity and a *Resource Repository or Organizational Model*, meaning that the *Participant* declaration may refer organizational structure definitions outside the scope of the specification, but which may be used with the extensibility mechanisms provided by XPDL.

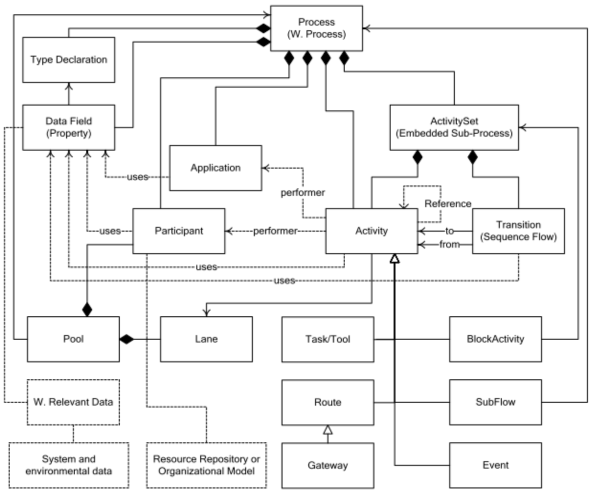

Figure 8 – Excerpt of the Process Definition Metamodel [22]

In XPDL, a *Process* is structured into *Activities*. The link between the active structure and *Activity*, is given by the *performer* relationship. The *Participant* identifiers that are used in this relationship must be declared either in the surrounding Process Definition or inherited from the surrounding *Package* declaration or coming from external packages, like an *Organizational Model*. The specification mentions the use of expressions to define the *Participants* of an *Activity*, without specifying exactly the syntax and semantics of these expressions. The specification also mentions that when the expression evaluation returns an empty set of performers or when it returns a non-unique performer, then this must be handled by the execution engine of the Workflow System and is outside the scope of the specification.

2.5 UML Activity Diagrams

UML is a standardized general-purpose language that aims "to provide system architects, software engineers, and software developers with tools for analysis, design, and implementation of software-based systems as well as for modeling business and similar processes" [10].

The modeling concepts of UML are grouped into language units represented by different diagrams, which consists of tightly-coupled modeling concepts that provide users the ability to represents aspects of a system under study according to a particular formalism. For instance, the Activity Diagram groups concepts related to behavior modeling.

UML 2.0 does not provide a specific language unit to model an organization; however, as shown on [4], general organizational structures can be modeled by UML class diagrams, and concrete organizations can be treated as instances of these general organizational structures.

Activity diagrams can also be used for process modeling in UML. An *Action* is one of the main constructs of an activity diagram, and a fundamental unit of behavior specification, taking a set of inputs and transforming them on a set of outputs (though either or both sets may be empty). An action represents a single step within an *Activity*, that is, one that is no further decomposed [10].

The connection of the active structure to the process models is done within an activity diagram using the notational element *ActivityPartition*, which divide the nodes and edges to constrain and show a view of the contained nodes. Constraints vary according to the type of element that the partition represents, which may be one of the following [10]: (i) *Classifier*, meaning that the behaviors of invocations contained by the partition are the responsibility of instances of the classifier representing the partition. Thus different instances of the same classifier may execute the contained actions; (ii) *Instance*, imposing the same constraints as a classifier-based partition, but requiring a particular instance of the classifier. (iii) *Part*, meaning that the behaviors contained in the partition will be executed by parts of the same instance of a structured classified. (iv) *Attribute a*nd *Value*, meaning that certain attributes are restricted to certain values. The specification includes an example of a partition representing a location attribute and sub-partitions representing specific values of that attribute, such as "Rio de Janeiro" [10]. Nevertheless, this latter kind of partitioning is not well documented in the specification, as it does not specify whether the attributes apply to actions inside the sub-partition or to objects (instances) executing the actions. Figure 9 exemplifies multidimensional partitioning.

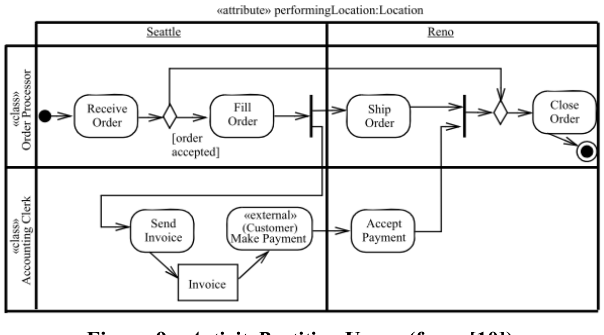

Figure 9 - *ActivityPartition* Usage (from [10])

The actions "Receive Order" and "Fill Order" in Figure 9 are performed by an instance of the "Order Processor" class, situated in "Seattle", but not necessarily the same instance for both. Although the "Make Payment" action is contained within the "Seattle/Accounting" partitions, its performer and location are not specified by these partitions since this action is stereotyped as «external».

2.6 BPMN

Business Process Modeling Notation (BPMN) is a standard graphical notation for business process modeling adopted by the OMG. Its main goal is to provide a notation that is easy to understand by all business users [23]. The BPMN 2.0 specification clearly states that the language is constrained to support only concepts of modeling applicable to business processes, meaning that other domains of an organization are out of scope, with one of them being the domain of organization modeling [9]. Although BPMN does not include elements for organizational modeling the specification clearly assumes the existence of these elements when defining who will be responsible for a process or for the execution of an activity.

BPMN defines a number of diagrams to model business processes under a certain perspective. We focus here on the Process and Collaboration Diagrams, not explicitly discussing Choreography (a specialization of Collaboration) and Conversation (a specialized use of Collaborations) [9].

The Process Diagram is used to model a business process internal to an organization. It essentially describes a sequence or flow of Activities in an organization with the objective of carrying out work. This type of diagram does not include a textual nor graphical way to explicitly specify the responsible for the *Process* or the activities contained within it. Nevertheless, *Lanes* can be used *informally* for that purpose. As discussed in the specification, "the meaning of the Lanes is up to the modeler" [9]. In practice, "Lanes are often used for such things as internal roles (e.g., Manager, Associate), systems (e.g., an enterprise application), an internal department (e.g., shipping, finance), etc." [9]. Figure 10 shows a small example of a *Process* defined in BPMN. *Activities* are represented by rectangles with rounded corners, and represent points in a *Process* where work is performed, being the main behavioral concept in the language. An *Activity* is an abstract metaclass specialized into either a *Sub-Process* or a *Task* (which in turn is further specialized into specific kinds of tasks).

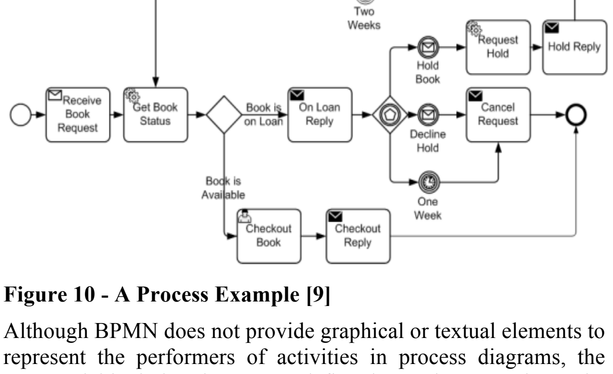

Figure 10 - A Process Example [9]

Although BPMN does not provide graphical or textual elements to represent the performers of activities in process diagrams, the metamodel includes elements to define them. Figure 11 shows the main concepts and associations related to this aspect of the language. The *Resource* metaclass is used to specify resources that may be referenced by activities. These resources may be human resources or any other resource assigned to an activity during process runtime. Resources are defined at type-level, e.g., “Professor” and “Student”. Specific resources (instances such as, e.g., “João Paulo” and “Rômulo”) would be described in a deployment phase, which is outside the scope of the specification [9], and may be addressed in a BPMN-conformant infrastructure. A modeler may characterize *resources* by defining its properties using *ResourceParameters* [9]. The assignment of active structure to behavior may be defined by the modeler using the *ResourceRole* element shown in Figure 11. The assignment may be done by defining either: (i) an association between the *ResourceRole* and a *ResourceAssignmentExpression* or (ii) between the *ResourceRole* and a *Resource*.

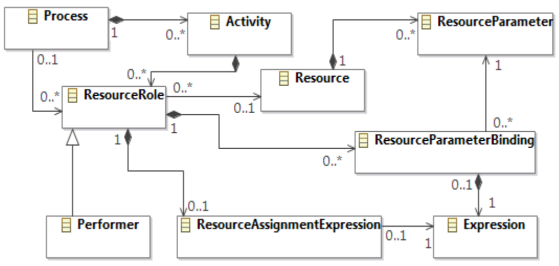

Figure 11 - Fragment of the BPMN metamodel centered in ResourceRole, adapted from [9]

In the former case (i), the modeler provides an *Expression* written in natural language or in a formal expression of choice (by default formal expressions are defined in XPath [2]). This expression is used at runtime to assign resource(s) to a *ResourceRole* element.

In the latter case (ii), a specific resource (type) is selected at modeling time. Optionally, the modeler may define which parameters of the resource specified may be used or overridden through the definition of an Expression, that may also use data of the instance task in which the resource is being referred.

Figure 11 shows that a *ResourceRole* may be further specialized in a *Performer*, meaning that the resources selected must be the ones responsible for the execution of the activity (“A performer can be specified in the form of a specific individual, a group, a position or role in an organization, or an organization” [9]).

In addition to the Process Diagram, BPMN defines a Collaboration to describe the interactions (messages exchange) between two or more business entities. These business entities are called *Participants* in the scope of a *Collaboration* and are represented graphically as pools. A *Participant* can be a specific entity (*PartnerEntity*, e.g. a company) or a more generic one (*PartnerRole*, e.g. a buyer). However, there are no graphical elements o distinguish these concepts, with all being done in natural language. A *Participant* may be associated with a *Process* in a *Collaboration*, meaning that it is responsible for the execution of the process. Figure 12 shows the metaclass *Participant* and its main associations.

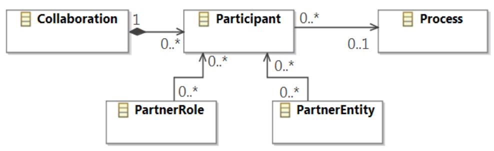

Figure 12 - Fragment of the metamodel centered in Participant, adapted from [9]

Figure 13 shows an example of a Collaboration Diagram. “Financial Institution” and “Supplier” are the *Participants*. Each one of them is assigned to a process.

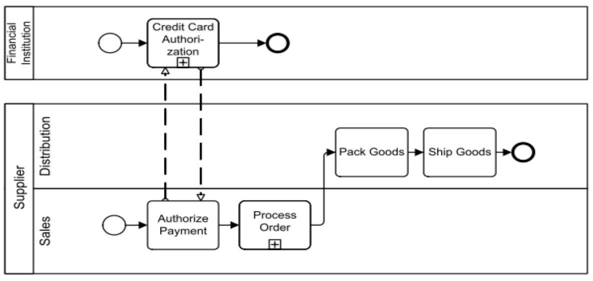

Figure 13 - A Collaboration Diagram (from [9])

If activities are represented in a collaboration, they may also be allocated to perfomers using the mechanisms discussed for the process diagram. We believe this may cause certain semantic problems in the language, because it allows modelers to mix activity-level assignment and process-level assignment with no consistency rules. (*Performer* is defined at activity level, i.e., a performer is assigned to an activity defined in a process, “being the resource that will perform or be responsible for an activity”, while *Participants* are defined at process-level. The metamodel does not define relations or constraints involving the metaclasses *Performer* and *Participant*.)

3. CONCLUSIONS AND FUTURE WORK

A mature approach to enterprise modeling should clearly establish relations between the various architectural domains addressed. In this paper, we have reviewed the mechanisms employed in ArchiMate, DODAF, ARIS, XPDL, UML and BPMN to support the assignment of active structure and behavior. We can observe in our analysis that most of the approaches offer simplistic support for this assignment, including few modeling elements to relate each of the architectural domains.

With respect to ArchiMate, *Business Actors* are assigned to *Business Behavioral Elements* indirectly, through the *Business Role* element. The language also includes a notion of *Business Collaboration* which may be used to assign a behavioral element to several *Business Actors* (through an aggregation of *Business Roles*). The objective of the language is to establish a high level abstract view on an enterprise architecture, and thus the language cannot be used to model details of the assignment.

DoDAF, in its turn, offers more expressiveness when considering the constraints on its assignment relation, defining *Conditions* under which the *Activity* should be performed and *Rules* on the *Performer*, possibly including quantitative constraints using a notion of *Measure*.

Regarding ARIS, we observed that it is the only one of the studied languages to define relationships beyond assignment or responsibility for behavior execution. The relations between active structure elements and behavioral elements include technical responsibility, participation in decision making, general contribution, general interest in, need to consult and inform, etc. Nevertheless, the semantics of each of the different relations is not discussed explicitly, and can only be superficially inferred from the names of the meta-associations.

Regarding XPDL, which is designed with the main goal to provide interoperability between workflow systems, the support for active structure assignment is rather primitive: it only identifies a direct relationship between a *participant* and an *activity*. XPDL makes no assumptions on the organization model (beyond defining a list of participant types, whose semantics is poorly defined.) The specification also mentions that expressions may be used to define the performers of activities, but a language for these expressions is not defined.

UML provides the generic mechanism of *ActivityPartitions* which can be used to define the classes or instances which execute actions in an activity diagram. The same mechanism can be used to capture any other criteria which modelers may define for grouping actions. The construct is similar to that of Lanes in BPMN, although specific stereotypes facilitate the identification of the types of partitions in a model, defining more precise semantics for each of them.

With respect to BPMN, the assignment of the performers may be done directly or through *expressions*. Differently from XPDL it provides a default language for such expressions. Nevertheless, it only assumes the existence of attributes in a (external) resource model. No kinds of relations between resources (performers) are assumed, and thus the *expressions* cannot take advantage of using relationships between active structure elements. Further, we have identified some issues in the combination of process-level and activity-level assignment relations. Some of the limitations in BPMN to address the assignment of active structure and behavior have been addressed in [1] and [7], which propose an extension to BPMN in order to support various kinds of active structure allocation proposed by [11].

We conclude that a complete integrated approach to the assignment of active structure and behavior is yet to be incorporated into the languages and frameworks considered here. We intend to address this gap in future work, by proposing language extensions and mechanisms that would allow the specification of assignment with different degrees of constraining, precise semantics and rich expressiveness, possibly using the creation resource patterns defined in [11]. We also intend to expand the scope of our investigation to include other enterprise modeling techniques and frameworks such as MODAF, UPDM and the RM-ODP Enterprise Language, and to identify expressiveness requirements that can be used to evaluate the techniques and frameworks systematically.

4. ACKNOWLEDGMENTS

This research is funded by the Brazilian Research Funding Agencies FAPES (grants number 45444080/09 and 52272362), CNPq (grants number 309059/2008-9, 481906/2009-6 and 483383/2010-4) and CAPES.

5. REFERENCES

[1] Awad, A., Grosskopf, A., Meyer, A., and Weske, M. *Enabling resource assignment constraints in bpmn*. Technical report, Business Process Technology – Hasso Plattner Institute, 2009.

[2] Clark, J., and DeRose, S. *XML path language (XPath) version 1.0*. Technical report, World Wide Web Consortium (W3C) Recommendation, 1999.

[3] Davis, R. *Business Process Modelling with ARIS - A Practical Guide*, Springer, 2001.

[4] Dumas, M., van der Aalst, W. M. P., and ter Hofstede, A. H. M. *Process Aware Information Systems: Bridging People and Software Through Process Technology*. Wiley-Interscience, 2005.

[5] Havey, M. *Essential Business Process Modeling*. O'Reilly Media, Inc., 2005.

[6] Lankhorst M., et al. *Enterprise Architecture at Work - Modelling, Communication, and Analysis*. Springer, 2005.

[7] Meyer, A. *Resource Perspective in BPMN: Extending BPMN to Support Resource Management and Planning*. Master's Thesis, Hasso Plattner Institute, 2009.

[8] Muehlen, M. Z. *Organizational Management in Workflow Applications – Issues and Perspectives*. Information Technology and Management, 5 (3-4), 271-294, 2004.

[9] Object Management Group (OMG). *Business Process Modeling Notation (BPMN) 2.0 Specification*. OMG document, formal/2011-01-03, 2011.

[10] Object Management Group (OMG). *The Unified Modeling Language: Superstructure*. *Version 2.3*, OMG document, formal/2010/05/03, 2010.

[11] Russell, N., ter Hofstede, A.H.M., D. Edmond, and van der Aalst, W.M.P. *Workflow Resource Patterns*. Technical report, Queensland University of Technology, Australia, 2010. http://www.workflowpatterns.com/patterns/resource, last accessed at 17/05/2011.

[12] Santos Jr., P. S., Almeida, J. P. A., and Pianissolla, T. L. *Uncovering the Organizational Modelling and Business Process Modelling Languages in the ARIS Method*. International Journal of Business Process Integration and Management (IJBPIM), Vol. 5, No. 2, 2011, 130-143.

[13] Santos Jr., P.S., Almeida, J. P. A., and Guizzardi, G. *An Ontology-Based Semantic Foundation for ARIS EPCs*, In: 25th ACM Symposium on Applied Computing (Enterprise Engineering Track), 2010.

[14] Scheer, A. W. *ARIS – Business Process Modeling (3rd edition)*, Springer, 2000.

[15] Shapiro, R. M. *XPDL 2.2: Incorporating BPMN 2.0 Process Modeling Extensions*, in Fischer, L. (ed.), 2010 BPM and Workflow Handbook, Future Strategies, Inc., 2010.

[16] Sharp, A., and McDermott, P. *Workflow Modelling Tools for Process Improvement and Application Development*. Artech House, 2001.

[17] Sowa, J. F., and Zachman, J. *Extending and formalizing the framework for information systems architecture*, IBM Systems Journal, vol. 31, no. 3, 1992, 590-616.

[18] The Open Group. *TOGAF Version 9*. 2009. http://pubs.opengroup.org/architecture/togaf9-doc/arch/, last accessed at 06/09/2011.

[19] The Open Group. *ArchiMate Technical Standard*. 2009. http://www.opengroup.org/archimate/doc/ts_archimate/, last accessed at 21/05/2011.

[20] US Department of Defense. *DoD Architecture Framework Version 2.02*,. http://cio-nii.defense.gov/sites/dodaf20/, last accessed at 25/05/2011.

[21] van der Aalst, W.M.P. *Business Process Management Demystified: A Tutorial on Models, Systems and Standards for Workflow Management*. LNCS, Vol. 3098, 2004, 1-64.

[22] WfMC, *Process Definition Interface – XML Process Definition Language (XPDL)*. WfMC Standards, WFMC-TC-1025, The Workflow Management Coalition, 2008.

[23] White S. A. *Introduction to BPMN*. IBM White Paper, 2004.

[24] Zachman, J. *A Framework for Information Systems Architecture*. IBM Systems Journal, Vol. 26, No. 3, 1987, 276-292.

APPENDIX: SUMMARY OF CONCEPTS AND THEIR RELATIONS

	Active Structure Domain		**Relations between Active Structure and Behavioural Domain**	**Behavioral Domain**
	Main Concepts	**Relations Between Concepts**		**Main Concepts**
ArchiMate 1.0	*Business Actor*, *Business Role*, and *Collaboration*	A *Business Actor* may be assigned to a *Business Role* and a *Business Collaboration* aggregates *Business Roles*.	A *Business Role* is *assigned to* a *Business Behavior Element*, with different semantics when used to relate different kinds of elements.	*Business Behavior Element*, specialized into *Business process*, *Business Function*, *Business Interaction* and a *Business Event*
DODAF 2.02	*Performer*, which may be a *System*, *Service*, *PersonType* or *OrganizationType*; and *IndividualPerformer*, which may be a specific *Organization* or *IndividualPerson*	*IndividualPerson* is instance of *PersonType*; *Organization* is instance of *OrganizationType*; part-whole relations	*activityPerformedByPerformer*, *Rule*, *ruleConstrainsActivityPerformedByPerformer*, *Condition*, *activityPerformableUnderCondition*, *Measure*, *measureOfTypeActivityPerformedByPerformed, measureOfTypeActivityPerformableUnderCondition*	*Activity*
ARIS	*Organization Unit Type*, *Organization Unit*, *Position*, *Person Type*, *Employee Variable*, *Person*, *Group*	Numerous relations, which are omitted here due to space constraints - see [12].	An element of the active structure domain may be related to a *Function* through these relations: *is technically responsible for*, *carries out*, *is IT responsible for*, *decides on*, *must be informed about, contributes to*, *accepts*, *has consulting role in*, *must be informed on cancellation* and *must inform about result of.*	*Function*
XPDL 2.1a	*Participant* may be a *Resource*, *Resource Set*, *Organizational Unit*, *Role*, *Human* or *System*.	None.	A *Process* includes the definition of *Participants*. The link between an *Activity* and a *Participant* defines the *performer* attribute, which may be defined using expressions. *Participant* identifiers are used in the *performer* attribute and must be declared in the surrounding process definition or coming from external packages, like an organizational model.	*Activity*, *Process*
UML 2.0 Activity Diagram	No specific elements, although active structure can be represented through *class diagrams* and *object diagrams* [4].	*Associations* in *class diagrams* and *links* in *object diagrams (*when these diagrams are used to represent active structure).	*Partitions* are contained within *Activities*, constraining and providing a view on the *Actions* performed in *Partitions*. May be used to indicate who/what will perform the *actions* contained within it, referring to an element such as *Classifier*, *Instance*, *Part*, *Attribute* and *Value*.	*Activity* and *Action*, which represents a single step within an *Activity*, that is, one that is no further decomposed.
BPMN 2.0	*Resource, PartnerRole* and *PartnerEntity.*	None.	A *Resource* may be associated to an *Activity* through the *Performer* metaclass. A *Performer* may explicitly specify a *Resource* who will perform the activity or an *Expression* that returns *Resources* that will perform the activity. A *Participant* of a *Collaboration* may refer to a *PartnerRole* or a *PartnerEntity* who will participate in the *Collaboration*. A *Participant* may also explicitly refer to a *Process*, which in turn contains *Activities.*	*Collaboration, Process, Activity*