A Framework for Business Process Modeling and Alignment

Artur Caetano, Technical University of Lisbon, Portugal, & INESC-INOV, Lisbon, Portugal; E-mail: artur.caetano@inov.pt
Marielba Zacarias, Technical University of Lisbon, Portugal, & INESC-INOV, Lisbon, Portugal
Pedro Sousa, Technical University of Lisbon, Portugal, & INESC-INOV, Lisbon, Portugal
António Rito Silva, Technical University of Lisbon, Portugal, & INESC-ID, Lisbon, Portugal
José Tribolet, Technical University of Lisbon, Portugal, & INESC-INOV, Lisbon, Portugal

ABSTRACT
This paper proposes an enterprise architecture framework that allows modeling organizational components for reuse and co-development, emphasizing their traceability and alignment at both design and execution time. It focuses on describing how entities interact in the context of business processes and relate to goals, actors and supporting systems. To facilitate the analysis of the architecture models, a set of five views that separate multiple organizational concerns is also introduced.

1. INTRODUCTION

Representing knowledge about an organization proves to be a challenging task since it requires several of its aspects to be represented in a coherent and integrated way. Failing to deliver such representation hinders the assessment of the organization, as well as the detection of problems and areas of improvement. For an organization to change it must be self-aware, meaning that if the knowledge on the organizational components is not shared and understood there will be a gap between the actual state of affairs and the state as perceived by the different stakeholders. In addition, information systems accentuate these issues as they facilitate information sharing and process automation, regardless of the quality of the information and how processes are aligned with the organization goals. Despite investments on systems and technology, organizations often do not have the adequate methods that enable the management and coordination of these systems to support planning, decision making, controlling and, especially, to leverage competitive advantage.

Enterprise architecture results from the process of representing and aligning the components that are required for the management of the organization. It is the set of representations required to describe a system or enterprise regarding its construction, maintenance and evolution (Zachman, 1987). It concerns the structure of the things of relevance in the enterprise, their components, and how these components fit and work together to fulfill a specific purpose within the organization. Identifying the architecture of the enterprise should therefore be considered as a fundamental step to understand and align the organizational components.

Extensive related work can be found on the literature. ANSA was likely the first project to propose views, claimed to provide complete coverage of information processing systems (ANSA, 1989; Herbert, 1994). The views on enterprise, information, computation, engineering and technology were later taken up in open distributed processing standards. The concept of view enables separating the multiple concerns of a system in such a way that they can be individually addressed and later composed in a global representation. Thus, this concept shares a common goal with other approaches to enterprise architecture.

RM-ODP (Farooqi, 1995; ISO, 1995; Schurmann, 1995) aimed at integrating and maintaining consistency between multiple distributed-systems standards. It includes descriptive elements that provide a common vocabulary and prescriptive elements, known as viewpoints, which constrain what can be built. Specifically, it defines the enterprise viewpoint for system boundaries, policies, and purpose; the information viewpoint to represent distributed information; the computational viewpoint for decomposition of system into distributable units; the engineering viewpoint for description of components needed and, finally, the technology viewpoint for describing the implementation details of components.

The Zachman Framework (O'Rourke, 2003; Zachman, 1987) is used both from modeling and management perspectives. It describes the subjects needed for developing and documenting the enterprise architecture in a matrix. The vertical axis defines multiple perspectives on the architecture while the horizontal axis offers a classification of its artifacts. Its rows are structured around the perspectives related to user roles, namely Scope, Enterprise Model, System Model, Technology Model and Detailed Representations, while the six columns focus on separating Data (who), Function (how), Network (where), People (who), Time (when) and Motivation (why). The framework is independent of specific methodologies, but does not define how to integrate the information within each cell, nor how to describe how to trace such information neither how to specify the artifacts within each cell (Frankel, 2003).

This paper proposes an enterprise architecture framework that emphasizes the traceability and alignment between organizational components, facilitating their reuse and co-development. It focuses on describing how entities interact in the context of business processes and relate to organizational goals, actors and supporting systems. The components of the proposed framework and its underlying UML representation (OMG, 2004) are presented in section 2. In order to facilitate the analysis of the architecture, section 3 describes a set of five views that separate organizational, business, information, application and technological concerns. Section 4 introduces the concept of context to enable aligning the designed components with their corresponding execution so that organizational self-awareness can be maintained. Finally, section 5 provides concluding remarks.

2. THE FRAMEWORK

The architecture defines and relates the fundamental concepts required to describe the enterprise in a set of blueprints. The current section describes these concepts as packaged UML classes. Section 3 describes the five views that encompass these packages.

An organization can be abstracted as a collection of business nouns that interact as described by verbs. The nouns represent the concepts within the organization that are of interest regarding the purpose of the model. The verbs are enterprise activities that define how work is done and how value is added, thus describing its business processes. These abstractions are modeled as entities, roles and activities. Entities (business nouns) display behavior by playing a number of roles. Activities (business verbs) specify how roles collaborate in order to achieve a given purpose.

2.1 Entity

An organization is composed of entities. An entity can be a person, place, machine, resource, event that has meaning in the context of the business and about which some information can be stored because it is relevant for the purpose of the model.

Entities can be classified, in the object-oriented sense, according to its attributes and methods. These features can be either intrinsic or extrinsic. Intrinsic features describe the entity in isolation, while extrinsic features arise from its relationships. For example, the entity "person" has intrinsic features such as age and sex, and extrinsic features such as job position and salary, which derive from a relationship between the person and its employer. The state of the intrinsic features may change over time but always characterize the object, whereas extrinsic features are only meaningful while a relationship is valid.

Entities may also relate structurally to other entities, as in the case when an entity is composed by other entities. An entity class may also be specialized. Entities may interact with other entities only by playing roles in the context of a specific business activity. An entity is represented as a UML Class.

2.2 Role

A role is the observable behavior of an entity in the scope of a specific collaboration context, representing its features when it collaborates with other entities in the context of an activity. An entity relates to zero or more role classes through the stereotyped «play» relationship. Each role represents a subset of its external or extrinsic features in the context of a specific collaboration as defined in a role model.

Roles aim at separating the different concerns that arise from the collaborations between the entities fulfilling an activity. A role may be bound to multiple entities. Binding a role to an entity means that a specific instance of that entity is able to express the behavior defined by the role. It also means that the attributes and method of the role will be part of the entity's feature set. A role is also a type and may be classified according to its features, so it can be generalized and aggregated as a class. Roles are described in role models that describe how roles are structured and how they collaborate in order to fulfill a task. The role model may also specify constraints. Roles are described as UML Classes.

The structural relationships between roles are shown on class diagrams. Role models are packages that comprise a class diagram to describe the role structure and a UML dynamic diagram to describe its collaborations. The class diagram depicts the roles and the role associations required to fulfill a task. It also describes any constraints or business rules that govern the role associations.

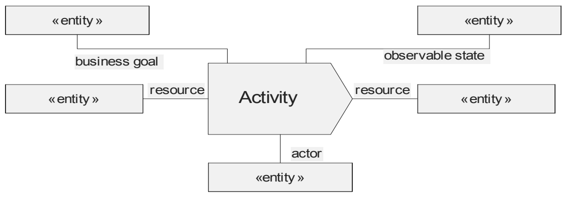
Figure 1 shows the structural dependencies between two roles, Employee and Employer, both defined in the "Works For" role model. It also depicts the binding between two entities, Person and Organization, and the two roles Employee and Employer. An activity is described by a number of role collaborations as seen on Figure 1 (right). This notation is closer to that of UML (OMG, 2004) and BPMN (BPMI, 2004).

The diagram in Figure 1 uses roles to separate the Person's external attributes from its intrinsic attributes. It can be observed that the job position and salary are extrinsic attributes and are dependent of the specific role Employee. Moreover, the role model makes clear that the Employee role relates with the Employer role, in the context of the "Works For" collaboration. Separating the intrinsic from extrinsic features allows entities to be designed independently of the activities that use them. This not only improves the reusability of the entities but also the ability to understand why a specific feature is expressed.

2.3 Activity

An activity is an abstraction describing how entities collaborate in order to produce a specific outcome. It aims accomplishing some task which, given an initial state, will always end in finite time and in a recognizable end-state. An activity may also be functionally decomposed into further activities. An activity specifies what entities are required to realize a task. As seen earlier, roles are used to separate the description of the actual entity features from the features required by the collaboration in context of the activity. In this way, activities and entities are described separately, and roles may be reused in different activities.

Figure 2. Common roles played by entities in the course of an activity



An activity often results from a number of interacting entities playing a set of roles specialized from four generic roles: resource, actor, observable state and goal (v. Figure 2). The resource role is played by the entities that are used as input to the activity. Resource entities are handled by a number of actors to generate output resources. An entity plays an actor role whenever is performing active behavior, such as entities modeling people, mechanical devices or information systems. During these operations, actors may contribute to the achievement of business goals.

From a methodological viewpoint, activities must relate to at least one entity playing the role of observable state. An observable state models a state of affairs that is of interest to a stakeholder. It can be seen as an indicator that results from performing the activity. This criterion can be used to decide if an activity can be further decomposed: decomposition is only meaning if all of its sub-activities produce at least one observable state. It is worth mentioning that the observable state set depends on the purpose of the architecture. For instance, the set of states in an architecture that will be used to identify information system requirements will likely be more fine grained detailed than the set used to describe the core activities of an organization from a strategic perspective. Observable states are detached from how activities are coordinated.

2.4 Activity Coordination

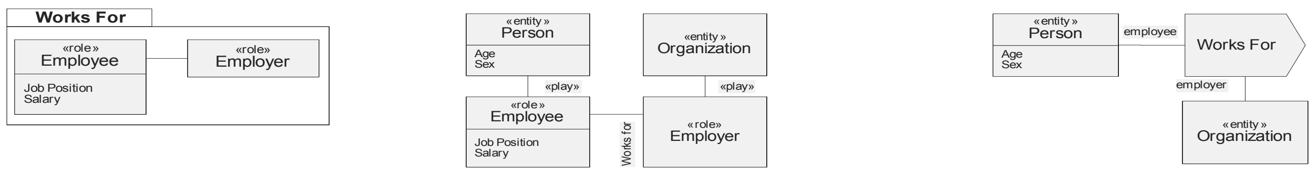
Coordination means linking together different parts of a system to accomplish a collective set of tasks. In the case of activity coordination, it means describing how activities are linked together so that they define a business process. The common definition of business process found in the literature defines it as a coherent collection of activities that takes one or more kinds of inputs and creates an output that is of value for an internal or external customer (Hammer, 2001; Verharen, 1997).

A process is coordinated and goal-driven. In this sense, a business process is a coordinated set of activities, but the converse may not be true. It is possible to describe activity coordination in different ways, such as using explicit control or data flow between activities or using events or pre-conditions. Activity coordination is represented using any of UML's dynamic diagrams, such as an activity diagram.

2.5 Role Types

The roles entities are able to play depend on the purpose of the model and on the specific organization. However, a number of basic roles are fundamental for organizational modeling:

Figure 1. The Works For role model showing the dependencies between two related roles (left); binding roles to entities (center); activity classifier (right)

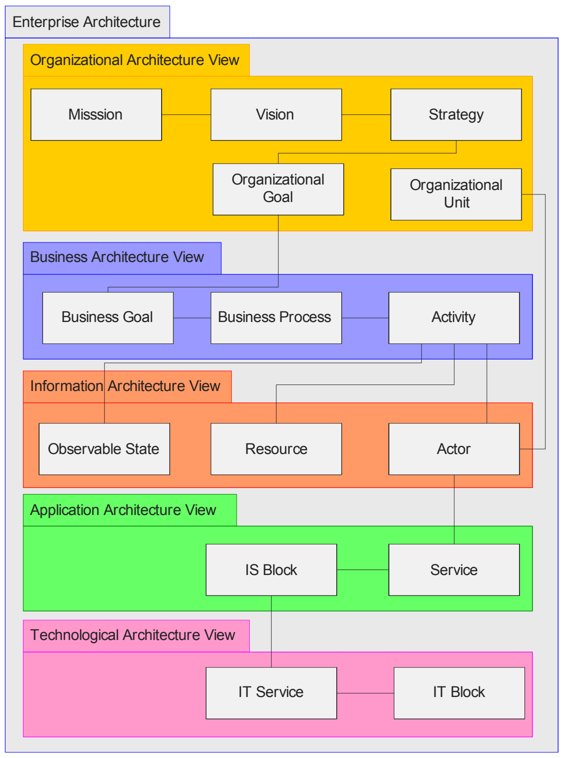


- **Mission**. A statement of enterprise's purpose.
- **Vision**. A statement on how to transform the mission into action.
- **Strategy**. A business process describing how to accomplish the vision.
- **Organizational Goal**. The goals achieved by the strategic process.
- **Business Goal.** A measurable state that the organization intends to achieve. Goals are achieved by entities involved in activity execution.
- **Resource**. The capacity of an entity to be managed by business process activities, including the ability of being consumed, incorporated, monopolized, or accessed.
- **Observable State.** A state of affairs that is of interest to a stakeholder in the context of the enterprise architecture. Observable states can guide the task of activity functional decomposing.
- **Actor**. An animate entity capable of active behavior. Actors model people, computer systems, mechanical tools or any other devices used to perform the operations required by an activity. Since entities only collaborate through roles, classifying an entity as an actor depends on the roles the entity is able to play, i.e., on the type of collaborations it participates in. This means that some entities may be potential actors but in a specific organizational case, they are just inanimate entities. The status of actor is transient and context dependent, meaning that the same entity could be an actor in the context of a process and a resource in the context of other. Actors are able to perform the set of services required to play a role. This means an actor is then responsible for providing such services. In case of people, these services are correlated to the skills, capabilities and other attributes pertaining to the person that are relevant to assign her to a role in the scope of an activity. In case of computerized systems or machines, the services represent the operations and functions that these devices put into play during the role assignment.

3. ARCHITECTURAL VIEWS

The architectural views aggregate and relate the fundamental roles played by the entities. They are defined to facilitate the analysis and development of the entities' roles through the separation of its different organizational concerns. Each view is individually represented and organized as a UML package that owns its model elements (v. Figure 3).

Figure 3. The enterprise architecture framework



3.1 Organizational Architecture View

This view deals with the aspects related with the organization but not to the specific business it conducts nor with the mechanisms used to accomplish the creation of value. It therefore includes concepts such as the enterprise mission, vision and strategy and the definition of organizational units.

3.2 Business Architecture View

The business architecture view synthesizes how business strategy is implemented and how processes are defined. The functional requirements of the business process support systems can be extracted from this view.

An activity describes the roles required for its operation. These roles are played by the organization entities and include actor role, resource role and observable state role. An activity requires one actor or a combination or team of actors to be executed. The actor represents a person, a machine or device, or an information system. An actor provides the services require for fulfilling the business role required by the activity. A resource is used as input or output of an activity during its operation. A resource is usually created, used, transformed or consumed during the operation of the activity. An observable state is specific resource role that is used as a means to observe the status of an activity. An activity is performed during a specific period. As a precondition for its enactment, all of the business roles must be fulfilled by specific entities. These entities will be engaged in playing their roles for the duration of the activity. The activity post condition is that all of the roles will have finished playing their part.

3.3 Information Architecture View

The information architecture describes what the organization needs to know to run its processes and operations as described in the business architecture. It defines a view on the business information that is system and technology independent. It is an abstraction of the information requirements of the organization and provides a high-level logical representation of all the key information elements that are used in the business as well as the relationship between them (Gilchrist, 2003; Inmon, 1999).

3.4 Application Architecture View

The application architecture view fulfills two goals: making explicit how business requirements are supported and allowing efficient management of the organization's entities. To satisfy these goals, the application architecture should be derived top-down from the analysis of the business and information architectures.

The application architecture defines the applications needed for data management and business support, regardless of the actual software used to implement the systems (Gilchrist, 2003). It functionally defines what application services are required to ensure processes and entities are supported in acceptable time, format and cost (Spewak, 1992). It describes the characteristics, styles and interactions among multiple applications. The architecture of a business process support system is described as a structure of Information System Blocks, each representing an organized collection of Services designed to handle organization information.

3.5 Technological Architecture View

The technological architecture view represents the technologies behind application implementation as well as the infrastructure and environment required for the deployment of the business process support systems. These concepts are abstracted as an Information Technology Block. An IT block realizes or implements IS blocks through number of technological Services.

4. ALIGNING DESIGN WITH EXECUTION

The concepts previously described enable capturing design-time aspects and assessing the static alignment between these aspects. Whereas this enhances the organization's self-awareness, it does not make explicit how to continuously and dynamically maintain this self-awareness. Aligning the organizational design with its execution entails capturing the current state of its active entities, i.e. actors and the particular interactions between them that change the state of resources, actors or activities. Actors interact using specific roles. Because of these interactions, actors continually change their own state, the state of activities and the state of resources.

A collaboration of resources and actors within an activity defines an interaction context that reflects the observable state of the activity, its actors and other resources. Actors are capable of engaging in multiple activities and to switch between them, potentially playing a different role on each. Several interaction contexts result from the different states of affairs related to the particular activities or roles where the actor is participates. When resuming a suspended activity the actors must be aware of the current state of affairs of the corresponding interaction context.

Figure 4 depicts the relationship between interaction contexts, actors, resources and activities. At design-time, the focus is on capturing goal, resource and actor roles related to an activity. However, during activity execution the observable features are actor interactions, which first create and then modify interaction contexts. These changes may trigger further changes on additional entities. In summary, modeling activity execution entails capturing (1) how actors interact, (2) how these interactions modify interaction contexts and (3) how interaction contexts trigger changes on the state of activities and other resources and actors. Defining models and modeling concepts to capture actual execution facilitates comparing the actual interactions of actors with activities and resources and the detection of errors or the discovery of emergent behaviors that improve the organization's effectiveness.

Figure 5 illustrates the interaction contexts and the relationship to design-time modeling concepts. The interactions captured are related to the collection of cards information for a mail application. These interactions resulted in two interaction contexts, the former shared by actors Peter and Maria and the latter, between Peter and the integration team. In the first context, Peter plays the role of task performer, Maria and the integration team provide resources for Peter. Successive interactions change the interaction context state, reflected by pending commitments. This interaction context modifies state of three activities.

Figure 4. Actors, resource, activities and interaction contexts

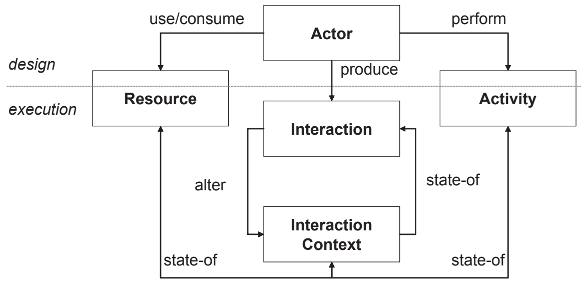
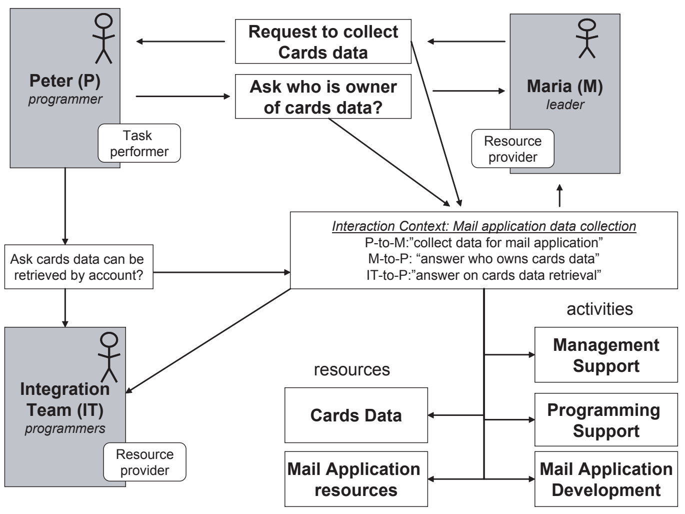


Figure 5. The "mail application data collection" interaction context


Peter (P)
programmer
Task performer
Request to collect Cards data
Ask who is owner of cards data?
Maria (M)
leader
Resource provider
Interaction Context: Mail application data collection
P-to-M:"collect data for mail application"
M-to-P: "answer who owns cards data"
IT-to-P:"answer on cards data retrieval"
Ask cards data can be retrieved by account?
activities
Management Support
Programming Support
Mail Application Development
resources
Cards Data
Mail Application resources
Integration Team (IT)
programmers
Resource provider

Interaction contexts are defined to capture execution. Identification and modeling of interactions contexts provides groupings of interactions exhibiting higher similarity with actual execution than other abstractions, such as activities. Interaction contexts relate to several activities and resources. Conversely, activities and resources relate to several interaction contexts. This means that interaction contexts are not part of activities or resources. Rather, they are a different concept. Moreover, actors activate their tasks according not only to task factors, but also to time, location, personal or inter-personal factors. Regarding interaction contexts as entities allows identifying interaction context emergent properties such as its priority and activation rules. This enables the discovery of actor scheduling heuristics and interaction rules. Analyzing interaction history makes also possible to find usage patterns and to discover tasks in a bottom-up fashion, facilitating the alignment between organizational models and actual execution (Zacarias, 2006).

5. CONCLUSIONS

Enterprise architecture consists of defining and understanding the different elements that shape an organization and how those elements are inter-related. This paper presents a framework for expressing the components of an architectural model for process-oriented organizations using five separate views that are integrated with the enterprise architecture model to facilitate its evolution. In this way, alignment becomes the process of continuously guiding the enterprise resources to exploit opportunities and cope with environmental changes.

The framework defines the fundamental concepts and their relationships. It also makes use of the object-oriented paradigm, exploiting mechanisms such as specialization and aggregation, with the goal of maximizing reusability and facilitating the discussion and communication of the models, thus promoting understandability.

REFERENCES

ANSA (1989). *ANSA Reference Manual, Release 01.00*. Architecture Projects Management Ltd., Cambridge,.

BPMI (2005). *BPMN 1.0 Specification*. http:// www.bpmn.org

Farooqi, K., Loggripo L., Demeere J. (1995). *The ISO Reference Model for Open Distributed Processing: An Introduction*. Computer Networks and ISDN Networks.

Frankel, D., Harmon P., *et al* (2003). *The Zachman Framework and the OMG´s Model Driven Architecture*. Business Process Trends.

Gilchrist, A. Mahon B. (2003). *Information Architecture: Designing Information Environments for Purpose*. Facet Publishing.

Hammer, M., Champy J. (2001). *Reengineering the Corporation: A Manifesto for Business Revolution*. Nicholas Brealey Publishing.

Herbert, A. (1994). An Overview of ANSA. *IEEE Network*. 8 January.

Inmon, W. (1999). *Data Architecture – The Information Paradigm*. QED Technical Publishing Group.

ISO (1995). *ISO/IEC 10746 ODP Reference Model*. International Standards Organization.

OMG (2004). *Unified Modeling Language: Superstructure, version 2.0, Revised Final Adopted Specification (ptc/04-10-02)*. http://www.omg.org/cgi-bin/doc?ptc/2004-10-02.

O'Rourke, C. Fishman N., Selkow W. (2003). *Enterprise Architecture Using the Zachman Framework*. ISBN 0-619-06446-3. Course Technology.

Schurmann, G. (1995). The Evolution from Open Systems Interconnection (OSI) to Open Distributed Processing (ODP). *Computer Standards and Interfaces*, Vol. 17.

Spewak, S. Steven H. (1992). *Enterprise Architecture Planning: Developing a Blueprint for Data, Applications and Technology*. Wiley-QED Publication.

Verharen, E, (1997). *A Language-Action Perspective on the Design of Cooperative Information Agents*. CIP-Gegevens Koninklijke Biibliotheek.

Zacarias M., Pinto H.S., Tribolet J. (2006). A Context-based Approach to Discover Multitasking Behavior at Work. *Task Models and Diagrams for User Interface Design (TAMODIA '06) International Workshop.*

Zachman, J. (1987). A Framework for Information Systems Architecture. *IBM Systems Journal*, vol. 26, no. 3, pp. 276-292.