ISeB (2007) 5:397–413
DOI 10.1007/s10257-007-0051-3

ORIGINAL ARTICLE

Enterprise modeling using the foundation concepts of the RM-ODP ISO/ITU standard

Alain Wegmann · Lam-Son Lê · Gil Regev · Bryan Wood

Published online: 30 May 2007

Abstract Enterprise architecture (EA) projects require analyzing and designing across the whole enterprise and its environment. Enterprise architects, therefore, frequently develop enterprise models that span from the markets in which the organization operates down to the implementation of the IT systems that support its operations. In this paper, we present SEAM for EA: a method for defining an enterprise model in which all the systems are systematically represented with the same modeling ontology. We base our modeling ontology on the foundation modeling concepts defined in Part 2 of ISO/ITU Standard ''Reference Model of Open Distributed Processing'' (RM-ODP). This work has two contributions to enterprise architecture: the SEAM for EA method itself and the use of Part 2 of the RM-ODP standard as a modeling ontology.

Keywords RM-ODP · Enterprise architecture · System modeling

1 Introduction

The alignment of business and IT is one of the top-ranked concerns for Chief Information Officers (CIO) (Luftman and McLean 2004). Enterprise architecture

A. Wegmann (✉) · L.-S. Lê · G. Regev
School of Computer and Communication Sciences,
Ecole Polytechnique Fédérale de Lausanne (EPFL), 1015 Lausanne, Switzerland
e-mail: Alain.Wegmann@epfl.ch

L.-S. Lê
e-mail: LamSon.Le@epfl.ch

G. Regev
e-mail: Gil.Regev@epfl.ch

B. Wood
Agile Enterprise Ltd, 11 Wilton Court, Sheen Road, Richmond TW9 1AH, UK
e-mail: bryan.wood@agile-enterprise.net

(EA) seeks to improve this alignment. Enterprise architects analyze and design systems that span from business entities (e.g. market, value network, business, department, employee) down to IT entities (e.g. IT systems, applications, software components, programming language classes). In doing so, they federate the views of the different disciplines involved in the design and operation of these entities.

Peyret (2004) classifies the different EA approaches into three categories:

- bottom-up approaches: architects focus on the inventory of the existing IT systems (traditionally called ''as-is'') and define governance rules for evolving these systems;
- top-down approaches: architects define the strategic goals for a company and derive the future IT-systems (traditionally called ''to-be'');
- change management approaches: architects analyze what is necessary in the existing IT systems and design the future IT systems necessary to support enterprise-wide projects.

SEAM (Wegmann 2003) is a family of methods for seamless integration between disciplines. SEAM for EA is an enterprise architecture method that belongs to the change management category. To simplify the discourse, we will use the term SEAM to designate SEAM for EA in the remainder of this paper. Enterprise architects can use SEAM, to develop an enterprise model, a model that represents the relevant features of the organization and its environment. These features depend on the nature of the project for which the model is necessary. They may span from the markets in which the organization operates down to the implementation of the IT systems that support its operations. The enterprise model is used by enterprise architects to document an existing organization and its environment, as well as to describe a future organization and its environment.

In this paper, we give an overview of the SEAM enterprise model and explain the ontology used to define the model elements and relations between them. This ontology is based on the foundation modeling concepts defined in Part 2 of the ISO/ITU Standard Reference Model of Open Distributed Processing (RM-ODP) (OMG 1995–1996). RM-ODP Part 2 was chosen as it gives us a concise and standardized definition (24 pages) of the necessary concepts needed to model systems. Thanks to these concepts we are able to systematically use one modeling technique for all systems, regardless of the nature of the system. Concretely, we use the same modeling technique to represent markets, value networks, companies, departments, IT systems, IT applications and software components. This is possible because we consider all of them as systems, which is the main originality of SEAM. SEAM is a graphical modeling language. Its notation has similarities with UML.[1] The notable differences are the capability to represent different kinds of information in a same diagram and to provide more contextual information. A SEAM modeling tool, SeamCAD, is in development (Lê and Wegmann 2005, 2006). All the illustrations of this paper were made with SeamCAD.

Our group has been developing SEAM for the last 6 years. Over this time, the key principles underlying SEAM have not changed—e.g. the RM-ODP ontology.

[1] OMG Unified Modeling Language, http://www.uml.org/

Prior work on SEAM related to RM-ODP has been published in (Lê and Wegmann 2005, 2006; Naumenko 2002, Naumenko and Wegmann 2007; Wegmann and Naumenko 2001; Wegmann 2003). SEAM has matured over the years and we now have concrete projects and courses in which it is used. Moreover, we have now concretely defined the necessary extensions to RM-ODP Part 2. These extensions have been validated by our practice and tools as well as through our on-going formalization effort performed with Alloy. The definition of these extensions together with their illustration in a concrete and detailed example are the core contributions of this paper.

In Sect. 2, we present an example of an enterprise model using SEAM. It introduces the SEAM model elements and notation. In Sect. 3, we discuss the applicability and the necessary extensions of RM-ODP Part 2 when used as an ontology for EA. In Sect. 4, we discuss the applicability and the future work we envision. In Sect. 5 we outline the related work.

2 Enterprise modeling with SEAM

We illustrate the SEAM modeling technique with an example of a company that works with its partner companies to sell products to its customers. Our enterprise model represents three kinds of systems: markets, value networks and companies. We consider that the company of interest, MarketingCo, has two partner companies ManufacturingCo and ShippingCo. These three companies define a Supplier Value Network (SVN). *Value network* (Stabell 1998) is a business term used to describe a group of companies that collaborate to create value for a customer. The Supplier Value Network together with the Customer make a *market* that we call ProductMarket.

In Sect. 2.1, we describe the service provided by the SVN to its Customers. We describe the Sell action of the SVN as a whole. By describing the SVN as a whole (as opposed to ''as a composite''), we purposely hide what each company does within the value network. By describing the Sell action (as opposed to the Sell activity), we also hide the details of the SVN behavior. The purpose of this representation is to show the net effect of the Sell action and of the collaboration among the companies that constitute the value network. It is then possible to reason on how different realizations of the Sell action and different configurations of companies can provide the same net effect.

In Sect. 2.2, we describe in more detail than in Sect. 2.1 the service provided by the SVN. The SVN is still considered as a whole. The Sell activity is modeled and this describes how the Sell action is realized. Representing the activities is useful for showing in detail the interaction between a system and its environment.

In Sect. 2.3, we describe the behavior of the companies that compose the SVN. The SVN is then considered as a composite. The purpose of this representation is to describe the responsibilities of each company. This representation is useful to reason about the implementation of a system.

In Sect. 2.4, we discuss the relations between the models presented in Sects. 2.1, 2.2 and 2.3. These relations are called functional—and organizational—level

traceability relationships. Understanding these relationships is important in order to verify the alignment between the business needs (e.g. the service provided by the value network, the individual company responsibility) and the IT implementation.

In this example, we model the Supplier Value Network in three different ways. We can model, using the very same modeling technique, how the departments of MarketingCo collaborate together and with the IT system, as well as how the IT system is built.

2.1 Model of an action in a working object considered as a whole

In SEAM, systems are represented as *working objects*. A working object captures both the behavior and the construction of a system. A working object can be considered as a whole or as a composite. When considered as a whole, we represent only the working object's behavior and the properties modified by the behavior. When considered as a composite, we represent the working object's components: these components are also working objects that can be represented as whole or composite.

Figure 1 represents three working objects: ProductMarket, SVN and Customer. ProductMarket is represented as a composite. SVN and Customer are represented as wholes. As working objects can represent different kinds of systems, SEAM allows for different pictograms to represent the different natures of the systems.

In Fig. 1, the SVN is represented as a whole. Therefore, only its behavior and its properties are shown. The rounded pictograms represent behavioral model elements and the angled ones represent the properties. All working objects have a Lifecycle

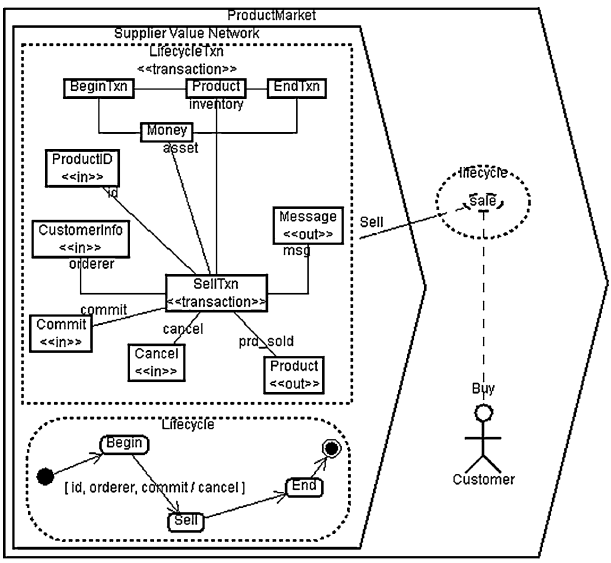

Fig. 1 SVN as a whole performs the Lifecycle activity (that includes the Sell partial interaction); Customer and SVN participate in the Sale full interaction

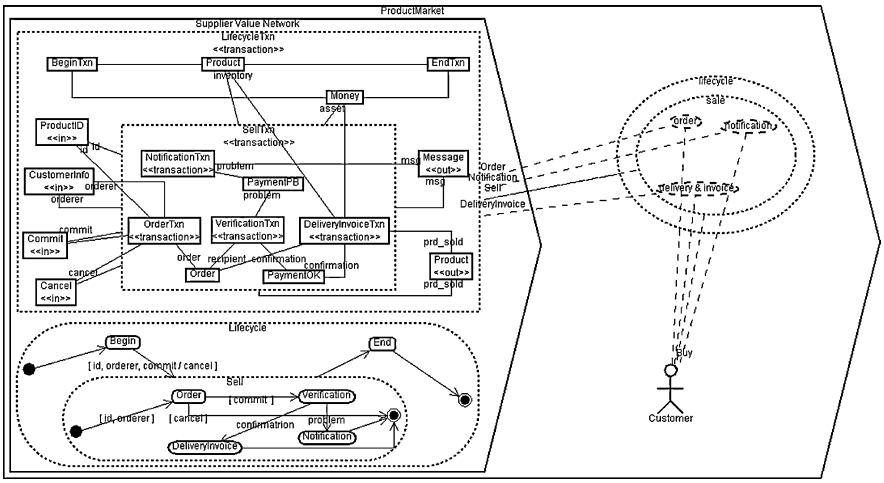

Fig. 2 SVN as a whole performs the Sell activity (that includes three partial interactions and the Verification internal action); Customer and SVN participate in three full interactions

activity that represents the working object behavior from its creation to its end. The SVN's lifecycle activity is composed of a Begin internal action[2] (corresponding to the initialization action, executed at the working object's creation), followed by a Sell partial interaction and an End internal action (corresponding to the termination action, executed at the working object's disappearance). For the properties, we represent transactions (stateless by definition) and stateful properties. Having transactions is useful for describing how stateful properties are used and modified by actions. In our example, the LifecycleTxn corresponds to the Lifecycle activity and the SellTxn transaction represents the occurrence of the Sell action. SellTxn shows that the Sell action modifies assets (of type Money) and inventory (of type Product), which are global properties (i.e. they exist during the whole LifecycleTxn). The Sell action also has id (of type ProductID) and msg (of type Message) as local properties (i.e. they exist only in SellTxn).

2.2 Model of an activity in a working object considered as a whole

Behavior can be represented in different levels of detail. For example, Fig. 2 represents the same working objects as Fig. 1. The difference is the behavior of SVN that is represented in more detail: the Sell action becomes the Sell activity.

In Fig. 2, the SVN's Sell activity is composed of the actions Order, Verification, DeliveryInvoice and Notification together with their execution constraints. The SVN's SellTxn transaction renders explicit the additional properties necessary to describe the Sell activity. For example, the Order property is useful for storing order related information necessary for the three actions: Order, Verification and

[2] The detailed definitions of the different kinds of model elements are in Sect. 3.

DeliveryInvoice. PaymentOK is the property that indicates that Verification was successful.

2.3 Model of an activity in a working object considered as a composite

Working objects can be represented as composite. Figure 3 represents the SVN as a composite (as opposed to a whole in Figs. 1, 2). It is then possible to understand how the companies that compose the SVN interact to perform the behavior described for the SVN as a whole.

In Fig. 3, each company that belongs to SVN is represented as a working object as a whole. Each company has its own behavior described with internal actions and partial interactions. The interactions of companies are represented by the model element called full interactions that exist between the companies. For example, the fact that ManufacturingCo provides a product to ShippingCo is represented by the

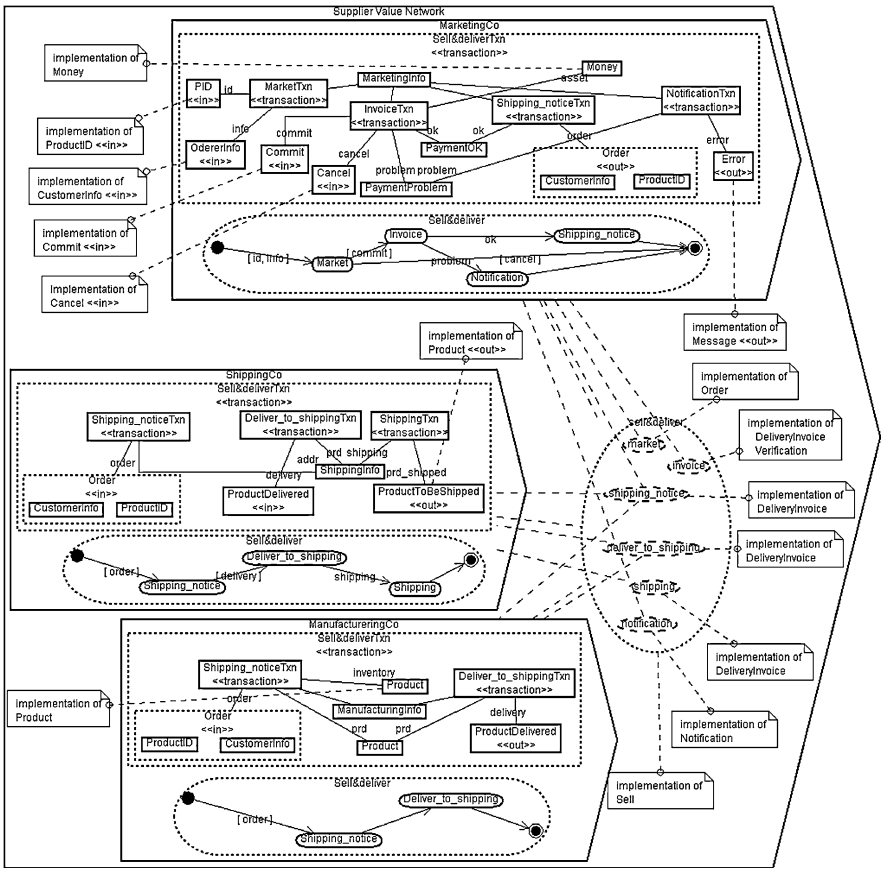

Fig. 3 SVN as a composite, composed of MarketingCo, ManufacturingCo and ShippingCo performing the Sell activity

partial interactions Shipping_to_Delivery found in each company. In ManufacturingCo, there is an output property called ProductDelivered and in ShippingCo, there is an equivalent input property. The fact that the two companies participate in this action is captured by the full interaction Delivery_to_Shipping (represented between both companies). This full interaction is realized by the partial interactions Delivery_to_Shipping executed by ManufacturingCo and by ShippingCo. The comment associated with the Delivery_to_Shipping full interaction indicates that it is the implementation of DeliveryInvoice specified for the SVN as a whole (see Fig. 2).

2.4 Functional and organizational traceability

In the preceding sub-sections, we have defined three representations of the SVN. The first representation includes SVN as a whole with the specification of the Sell action (Fig. 1). The second one also includes SVN as a whole, but with the Sell activity (Fig. 2). The third represents SVN as a composite together with the Sell activity (Fig. 3). Obviously there are relationships between these three representations of SVN. They are not independent. They correspond to what we call the traceability relationships. We define the *traceability relationship* as the behavioral equivalence between two specifications of model elements that the modeler wishes to consider as directly related (Wegmann et al. 2005a). For example, there is a traceability relationship between the SVN's Sell action (Fig. 1) and the SVN's Sell activity (Fig. 2). We call this kind of traceability: *functional traceability* or *traceability across functional levels*. The SEAM notation makes this relation explicit by keeping a reference to the Sell action when the Sell activity is represented. Furthermore, the actions that are represented in the Sell activity in SVN as a whole (Fig. 2) are directly related to the full interactions visible in the SVN as a composite (Fig. 3). The comments visible in Fig. 3 make this link explicit. The relationships between a behavior of a working object as a whole and the equivalent behavior of the ''same'' working object as a composite correspond to the *organizational traceability* or *traceability across organizational levels*.

Rendering explicit the traceability between functional levels and organizational levels is important for the verification of the alignment between business and IT. In SEAM we verify business and IT alignment by checking that we have all the necessary traceability relationships between the specifications of the value networks down to the specification of the IT systems. Having all traceability relationships is a necessary condition for business and IT alignment, but it is not a sufficient condition. For example, non-functional properties should also be verified.

3 The applicability of RM-ODP Part 2 to enterprise architecture

One of the challenges when designing a modeling technique is the modeling ontology. An ontology defines the terms used to build a model and the relations between these terms. In this section, we discuss the applicability and the necessary extensions of RM-ODP Part 2 when used as an ontology for EA.

The Reference Model for Open Distributed Processing (RM-ODP) is an ISO/ITU standard (OMG 1995–1996). RM-ODP defines a modeling infrastructure for distributed IT systems within organizations. The RM-ODP standard is composed of four parts. Part 1 is an overview of RM-ODP and is non-normative. Part 2 defines the fundamental concepts needed for modeling Open Distributed Processing systems. Part 3 presents an application of Part 2 for particular viewpoint specification languages (i.e. enterprise, information, computational, engineering, technology viewpoints). Part 4 is a partial formalization of the previous parts.

RM-ODP is known especially for its Part 3 that defines requirements for viewpoint languages useful to describe an IT system and its environment (Lankhorst 2005; Putman 2000). For example, the enterprise viewpoint is useful for describing the enterprise in which the IT system will be deployed; the information viewpoint is useful for describing the IT system specification; the computational viewpoint is useful for describing the computing structure of the IT system; the engineering and technology viewpoints are useful for the implementation of the IT system. All these viewpoints refer to the terminology defined in RM-ODP Part 2 (e.g. object, state, action, activity, type, instance).

Our approach is original because it does not rely on the RM-ODP viewpoints. These viewpoints describe the different aspects necessary to model an IT system. Each viewpoint has its own modeling language. In our approach, the goal is to have the same modeling language regardless of the subject to be modeled (e.g. business or IT systems) and to have a relatively small set of heuristics for the specific aspects of each subject. Hence, we base our work directly on RM-ODP Part 2 and we systematically use the concepts defined in RM-ODP Part 2 to represent systems that span from business down to IT (Wegmann and Naumenko 2001).

RM-ODP Part 2 first defines the *basic interpretation concepts*. These concepts are necessary to relate the universe of discourse to the model and to define the model elements. RM-ODP Part 2 then defines the *basic modeling concepts* (e.g. object, action, activity) and the *specification concepts* (e.g. type, instance). These are the concepts necessary to fully specify the model elements. In the following subsections, we describe in more details the Basic Interpretation Concepts and the Basic Modeling Concepts. Specification Concepts are not discussed in this paper as they are fully compatible with our approach.

3.1 Basic interpretation concepts

The *basic interpretation concepts* are necessary for understanding how the RM-ODP standard is constructed. When a model is created, relevant entities in the universe of discourse are represented as model elements. Model elements are defined by one basic modeling concept and one or more specification concepts. RM-ODP defines clauses that describe these concepts but does not explicitly state how to combine them. Naumenko (2002) and Naumenko and Wegmann (2007) propose to use Russel's Theory of Type and Tarski's Declarative Semantics to combine these concepts. We use the definition of the Sell action as one example of this combination. According to Russel's theory of type, we first take a model element that is considered as *something destitute of complexity*. Then, a first-order

predicate (corresponding to the basic modeling concept) is applied to this model element. This specifies the essence of the model element. In this example, the first order predicate defines the model element as an ''action''. The meaning of ''action'' is defined using Tarski's Declarative Semantics. RM-ODP Part 2 states that an action is *something which happens*. We consider ''*something which happens*'' to be the conceptualization of the universe of discourse that all modelers need to agree on. Once a model element is associated with a basic modeling concept, higher-order predicates (corresponding to the specification concepts) are applied. This is again an application of Russel's Theory of Type. With these higher-order predicates the model element that represents an action is ''specialized'' to represent the Sell action. All model elements are defined in this way (although some elements do not have an explicit conceptualization).

In RM-ODP Part 2 the terms *abstraction* and *atomicity* are defined. It is specified that *fixing a given level of abstraction may involve identifying which elements are atomic*. SEAM has two different kinds of levels of abstraction: functional and organizational. In the functional hierarchy, the element that determines the level of abstraction is the action (e.g. actions Begin, Sell and End in Fig. 1). In the organizational hierarchy, it is the working object the modeler considers as a whole that determines the level of abstraction (e.g. Supplier Value Network in Fig. 1 or MarketingCo/ShippingCo/ManufacturingCo in Fig. 3). Each hierarchy is composed of levels.

These notions of abstraction levels have their roots in constructivism: constructivism states that all knowledge is relative to the observer (LeMoigne 1995; Checkland and Scholes 1990). Observer-independent descriptions of reality do not exist. Different functional and organizational levels correspond to the various abstractions that the different kinds of observers have developed to simplify their understanding of systems. It so happens that these abstractions appear hierarchical and this is why we call them hierarchies. The functional hierarchy is frequently made explicit in system design, whereas the organizational hierarchy is made explicit more rarely, as people consider it obvious. This lack of explicitness can lead to misunderstandings as people often use the same term to designate different entities. Our organizational levels are inspired by the concept of organizational hierarchy defined by James Greer Miller in Living System Theory (Miller 1995). Miller has shown that a living system can be modeled systematically and hierarchically (from organization, made of groups, made of humans, made of organs, made of cells). We transpose Miller's approach to the enterprise context.

3.2 Basic modeling concepts

RM-ODP Part 2 defines basic modeling concepts such as object, action, and state. To directly support system modeling with RM-ODP Part 2, we had to define a few more concepts than those in the standard. We present these concepts in this section.

Our goal is to model systems. As defined in RM-ODP a system is *something of interest* seen *as a whole or as comprised of parts.* We consider the concept of system as an agreed conceptualization between the modelers. We define the *working object* as the model element that corresponds to the system conceptualization.

Working object is a specialization of the concept of *object* defined in RM-ODP: The original *object* is not associated with the system conceptualization. In our example, SVN, Customer, ProductMarket, MarketingCo, ManufacturingCo and ShippingCo are all working objects. This means that they are all perceived as systems in the universe of discourse.

We have also refined the definition of the different kinds of actions. In RM-ODP, actions are divided into internal actions and interactions. In RM-ODP Part 2, it is written: *the set of actions associated with an object is partitioned into internal actions and interactions.* To model systems as we propose, we need two kinds of interactions: full and partial. A *partial interaction* is an action of one working object of interest (represented as a whole) and involves one or more working objects in its environment. A *full interaction* is an action of one working object of interest (represented as a composite) and involves one or more of its component working objects and it may or may not involve working objects in the environment of the working object of interest. In Fig. 4, actions M_S, R_A and TinR_A are full interactions; actions M_A, M_B, R_C, R_D, R_E and TinR_C are partial interactions and action U_C is an internal action. Finally, we have introduced concepts necessary to structure the state space. RM-ODP Part 2 defines the concept of state as, *at a given instant in time, the condition of an object that determines the set of all sequence of actions in which the object can take part*. Our goal is to describe the state at the same level of detail as the behavior. For this reason, it was important to add a means to structure the state. This is the concept of property. Properties can be stateless or stateful. Stateless properties represent occurrences of actions. Stateless properties are called *transactions*. They are similar to the stateless objects presented in Bernardeschi et al. (1997). One special transaction is the lifecycle transaction that represents the overall working object lifecycle. Transactions are useful for representing the context in which stateful properties exist. Stateful properties store the system's state. They are similar to UML attributes except that they can be hierarchical (properties can be composite as well). Global properties exist in the context of the system lifecycle. They are created at the system's initialization and disappear at the system's termination. Local properties exist in the context of transactions with a shorter lifespan than the lifecycle transaction.

In summary, RM-ODP Part 2 is well adapted as an ontology for enterprise models but it would require a few extensions to be perfectly suitable. The extension we propose consists in the way we model systems with objects, in the definition of the full and partial interactions and in the concept of properties to structure the state.

4 The applicability of SEAM and future work

SEAM focuses on the functional and organizational analysis of enterprises, their environments, and of their IT systems. SEAM offers only a partial view of an enterprise: analyzing functionality across organizational levels is only a subset of what needs to be analyzed when designing an enterprise. For example, specialists might focus on non-functional properties (such as performance, security or finance).

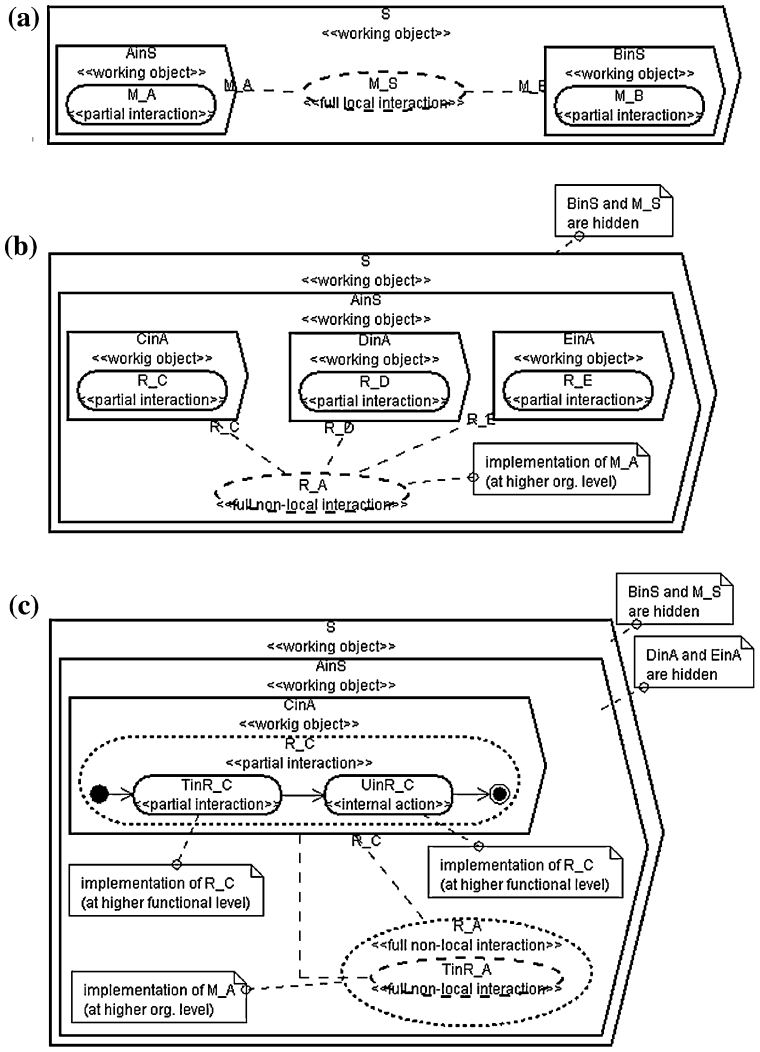

Fig. 4 Examples of actions and traceability relations between organizational levels (**a**, **b**); between functional levels (**b**, **c**)

Our experience shows that modeling functions across organizational levels adds value because it defines a common, minimal, understanding across the whole organization. To illustrate the use of SEAM, we provide three examples of how we have applied it in concrete situations:

- A mid-size IT-based company (approximately 200 employees) had to streamline its IT organization across product lines. This was a 5-year project that involved the whole company and multiple consultants. SEAM was successfully used to represent the roles of the company in its market, the roles of the company's

departments and the way the business processes needed to be structured. The benefits were the development of a standardized terminology and of a visual model that can be used by the CIO in his decision making process. This project is described in Wegmann et al. (2005b).

- A software company won a contract for a relatively large development of a tax management IT application for a public administration (approximately 4 years, 20 developers). SEAM was successfully used to model the project organization and the IT application in parallel with the RUP design process (Kruchten 2004). The goal for the SEAM model was to accelerate the training of new software developers. An on-line tutorial was developed, using the SEAM models as its main user interface. As a result, from an experiment with two developers, the training time was reduced from 6 to 2 weeks.
- A university was constructing a new building and the project goal was to equip this new building (furniture, multi-media systems, IT systems). This was an 18 months project with a 5M CHF equipment budget. A SEAM model was developed to build the business case and to specify to the vendors the equipment that was needed. SEAM was also used to generate a complete IT specification aligned with the project business specification. The project was successful.

In all these projects, the SEAM model was useful for reaching an agreement on what systems exist and on what functionality is needed. Once this was agreed upon, the different specialists had fewer difficulties in communicating with each other and used their common understanding to develop their own models. This illustrates why SEAM adds value, even if it enforces a hierarchical vision of the enterprise. The SEAM model is only considered as a shared model that all specialists can refer to while developing their own models. Developing the SEAM model does not prevent the specialists from using non-hierarchical models to reason about specific aspects.

Our future research work has two main directions: further evaluation of SEAM with additional projects and a more formal definition of its semantics. For this, we are currently running three projects: (1) formal definition of static, dynamic and invariant schemas in SEAM (Wegmann 2003)—similar to the schemas defined in RM-ODP Part 3. Our schemas have declarative semantics based on Alloy (Jackson 2002). (2) Behavioral simulation and alignment checking (Wegmann et al. 2005a) (with an operational semantics based on ASML, Börger and Stärk 2003); (3) synthesis of the results of (1) and (2) in a formal Alloy model of the SEAM ontology. This Alloy formal model will be automatically translated into Java code used in the SEAM tool.

5 Related work

In this section, we compare SEAM with existing EA frameworks and business process modeling techniques (Sect. 5.1), software and system-engineering methods (Sect. 5.2) and existing RM-ODP based approaches (Sect. 5.3).

5.1 EA frameworks and business process modeling

Our analysis shows that most frameworks do not provide a modeling ontology such as the one described in this paper. For example, Zachman (1987) (often considered as the first EA framework) and The Open Group Architecture Framework (TOGAF),[3] one of the most widely used frameworks, propose ad-hoc modeling frameworks.

Business process modeling techniques have, in general, some kind of ontology (necessary for simulating and executing). None of them are based on RM-ODP. In general, there are three main differences between SEAM and the business process modeling techniques. First, SEAM provides more contextual information than other business process modeling techniques (e.g. explicit modeling of behavior and properties, modeling of the system's environment, etc.). Secondly, SEAM proposes a systematic way to address business, business process and software modeling—aspects that are usually not as integrated in the other business process modeling techniques. Third, the existing business process modeling techniques provide more tool support (e.g. workflow management) than SEAM. We provide below examples of well-known business process modeling techniques.

The Computer Integrated Manufacturing Open System Architecture (CIMOSA, also known as the ISO EN/IS 19440 standard) focuses on the modeling of processes in the context of computer integrated manufacturing projects (Vernadat 1996). CIMOSA proposes a way to model processes at different levels of abstraction. This is similar to SEAM functional levels. However, CIMOSA does not have explicit organizational levels as SEAM does.

IDEF[4] (Integrated DEFinition Methods) is a set of methods that address many aspects of enterprise modeling (function, data, process, object-oriented design, ontology). SEAM proposes similar features but based on RM-ODP-based ontology. IDEF does not propose a concept equivalent to organization levels. Balabko et al. (2005) shows a comparison between IDEF0 and SEAM.

Business Process Modeling Notation (BPMN)[5] provides business users with a rich notation for modeling business processes. The processes defined in BPMN are hierarchical. BPMN can be translated to Business Process Execution Language[6] (BPEL) for workflow execution. Nevertheless, BPMN doesn't address business issues as SEAM does with the organizational levels. SEAM does not provide workflow execution.

Design and Engineering Methodology for Organizations (DEMO) is a method for (re)designing organizations (Dietz 2006). The DEMO ontology is rooted in the Communicative Action Paradigm. DEMO defines three types of models of an organization: the black-box model, the white-box model, and the flow model. The black-box model deals mainly with the external behavior of a system and supports

[3] The Open Group Architecture Framework (TOGAF), http://www.opengroup.org/togaf

[4] Integrated Definition Methods, http://www.idef.com/

[5] OMG Business Process Modeling Notation, http://www.bpmn.org/

[6] Web Service Business Process Execution Language, http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v20-rddl.html

the functional refinement. In the flow model, a system is conceived as a network of nodes transforming the input flows into output flows. The white-box model defines the constructional refinement of the system. It specifies the definition of subsystems (Dietz 2006). The main difference between SEAM and DEMO is the ontology used (RM-ODP instead of Communicative Action Paradigm).

It is also worth highlighting an approach that extend the notion of software component to the business: Turowski (2002) defines the concept of business components. He proposes seven levels of specification for this kind of component. These levels are market, task, terminology, quality, coordination, and behavioral levels. Most of these levels have their equivalent in SEAM (with the quality level as an exception).

5.2 System and software engineering

Numerous methods have been developed for hierarchical modeling in system engineering and in software engineering. We describe here some of the approaches that we consider similar to SEAM.

Object-Process Methodology (OPM) addresses the modeling of systems in general (Dori 2002). It has its own notation and provides a modeling tool called OpCat (Dori et al. 2003). SEAM differs from OPM mainly by its RM-ODP-based ontology. OPM was developed for modeling software systems and can be used to model enterprises. SEAM was designed to model enterprises and can be used to model software systems. The notations reflect these different approaches.

Catalysis (D'souza and Wills 1999) is a development process that analyzes and designs in three levels: business, IT system and software components. It uses its own UML-inspired notation. SEAM was inspired by Catalysis. The goal for SEAM is to provide a design method analogous to Catalysis, but with a broader scope (from business down to IT) and based on RM-ODP.

Systems Modeling Language (SysML)[7] is developed by the OMG. It is based on UML. SysML targets the design of large industrial systems (e.g. aircraft, power plants, etc.). SysML can model the context of the system to develop as well as the system itself.

KobrA (Komponentenbasierte Anwendungsentwicklung) (Atkinson et al. 2001) proposes a recursive model that describes IT systems/components. KobrA is based on UML. Both KobrA and SysML differ from SEAM by their tight link to the UML meta-model (as opposed to RM-ODP). Even if both methods can model multiple systems, they are designed to focus mainly on one system of interest.

5.3 ODP-related approaches

To our knowledge, the SEAM approach, which directly uses the RM-ODP Part 2 concepts, is unique. Other approaches are based on RM-ODP viewpoints as defined in Part 3. For example, Lupu et al. (2000) define viewpoint languages.

[7] OMG System Modeling Language, http://www.sysml.org/

Dijkman et al. (2004), Boiten et al. (2000), Dustzadeh and Najm (1997), Bernardeschi et al. (1997) check consistency between viewpoints. Romero and Vallecillo (2005) map viewpoints to UML, Steen et al. (2004) develop EA tools based on viewpoints.

It is important to mention the ISO/IEC 15414 enterprise language standardization[8] that refines and extends the enterprise language as defined in RM-ODP Part 3. This standard addresses enterprise modeling. SEAM has similar concepts to those defined in this standard. For example, we can consider the notion of working objects as similar to the concept of community object. Nevertheless, the 15414 standard uses extensively policies expressed with deontic logic, a feature not provided by SEAM.

The UML Profile for Enterprise Distributed Object Computing (EDOC)[9] embodies to some degree the RM-ODP Part 3 approach to system modeling. This profile is composed of seven standards (overview, meta-model for Java and Enterprise Java Beans (EJB), flow and collaboration specifications, pattern and relationship and relation to Meta-Object Facility (MOF)). The UML Profile for EDOC and SEAM share the same goal: model an enterprise. The main difference is in the ontology selected to express the models. In the UML profile for EDOC, their goal is to be as close as possible to UML and so they use the UML meta-model as ontology. In SEAM, our goal is to have an ontology as simple as possible, so we stay close to RM-ODP Part 2 (which is much simpler than the UML meta-model). A comparison between the UML meta-model and SEAM ontology can be found in (Naumenko and Wegmann 2002).

6 Conclusion

Enterprise architects need to develop enterprise models in order to describe an existing company or an expected change in a company. Enterprise models represent systems from business down to IT. In this paper we have presented SEAM, a method for developing such enterprise models. More importantly, we have shown how the ITU/ISO Standard RM-ODP Part 2 is applicable (with a few extensions) as an ontology for EA methods. This work contributes to linking the RM-ODP and EA communities by showing how RM-ODP Part 2 can be directly used for enterprise modeling.

The originality of our approach is that we bring together generic modeling techniques for all systems, as well as domain-specific heuristics (e.g. marketing heuristics for marketing or governance rules for IT). We have also briefly discussed some of the applications of the SEAM enterprise models. These models are not designed to replace discipline specific models but they complement them and they help federate multi-disciplinary teams.

[8] RM-ODP Enterprise Language, http://www.joaquin.net/cuml/Ent/index.html

[9] UML Profile for EDOC, http://www.omg.org/technology/documents/formal/edoc.htm

References

Atkinson C, Paech B, Reinhold J, Sander T (2001) Developing and applying component-based model-driven architectures in KobrA. In: Proceedings of 5th International EDOC Conference, Seattle, USA, 212–223, IEEE

Balabko P, Wegmann A, Ruppen A, Clément N (2005) Capturing design rationale with functional decomposition of roles in business processes modeling. Softw Process Improv Pract 10(4):379–392

Bernardeschi C, Dustzadeh J, Fantechi A, Najm E, Nimour A, Olsen F (1997) Transformation and consistent semantics for ODP viewpoints. In: Proceedings of FMOODS'97, Canterbery, UK

Boiten E, Bowman H, Derrick J, Linington P, Steen M (2000) Viewpoint consistency in ODP. Comput Netw 34(3):503–537

Börger E, Stärk R (2003) Abstract state machines: a method for high-level system design and analysis. Springer, Heidelberg. isbn 3-540-00702-4

Checkland P, Scholes J (1990) Soft system methodology in action. Wiley, Chichester. isbn 0-471-92768-6

Dietz J (2006) Enterprise ontology: theory and methodology. Springer, Heidelberg. isbn 3-540-29169-5

Dijkman R, Quartel D, Pires L, Sinderen M (2004) A rigorous approach to relate enterprise and computational viewpoints. In: Proceedings of 8th International EDOC Conference, California, USA, pp 187–200, IEEE

Dori D (2002) Object-process methodology, a holistic systems paradigm. Springer, Heidelberg. isbn 3540654712

Dori D, Reinhartz-Beger I, Sturm A (2003) OPCAT—a bimodal CASE tool for object-process based system development. In: Proceedings of 5th ICEIS, Angers, France

D'souza DF, Wills AC (1999) Object, components and frameworks with UML, the catalysis approach. Addison-Wesley, Reading. isbn 0-201-31012-0

Dustzadeh J, Najm E (1997) Consistent semantics for ODP information and computational models. In: Proceedings of FORTE/PSTV'97, Osaka, Japan

Jackson D (2002) Alloy: a lightweight object modelling notation. ACM Trans Softw Eng Methodol 11(2):256–290

Kruchten P (2004) The rational unified process: an introduction. Addison-Wesley, Reading. isbn 0-321-19770-4

Lankorst M (2005) Enterprise architecture at work. Springer, Heidelberg. isbn 3-540-24371-2

Lê LS, Wegmann A (2005) Definition of an object-oriented modeling language for enterprise architecture. In: Proceedings of 38th Hawaii International Conference on System Sciences, Hawaii, USA, 222a–222a, IEEE

Lê LS, Wegmann A (2006) SeamCAD: object-oriented modeling tool for hierarchical systems in enterprise architecture. In: Proceedings of 39th Hawaii International Conference on System Sciences, Hawaii, USA, 179c–179c, IEEE

LeMoigne JL (1995) Les épistemologies constructivistes. Que sais-je? Presses Universitaires de France. isbn 978-2130469438

Luftman J, McLean ER (2004) Key issues for IT executives. MIS Quarterly Executive 3

Lupu E, Sloman M, Dulay N, Damianou N (2000) Ponder: realising enterprise viewpoint concepts. In: Proceedings of 4th International EDOC Conference, Makuhari, Japan, pp 66–75, IEEE

Miller JG (1995) Living systems. University of Colorado Press. isbn 0070420157

Naumenko A (2002) Triune continuum paradigm: a paradigm for general system modeling and its applications for UML and RM-ODP, PhD thesis, School of Computer and Communication Sciences, EPFL

Naumenko A, Wegmann A (2002) A metamodel for the unified modeling language. In: Proceedings of <<UML>> 2002, Dresden, Germany, pp 2–17, Springer, Heidelberg

Naumenko A, Wegmann A (2007) Formalization of the RM-ODP foundations based on the triune continuum paradigm. Comput Stand Interfaces 29(1):39–53

OMG (1995–1996) ISO/IEC 10746-1, 2, 3, 4 | ITU-T Recommendation, X.901, X.902, X.903, X.904, Reference Model of Open Distributed Processing

Peyret H (2004) Getting Value From Enterprise Architecture Tools, Market Overview, Forrester

Putnam JR (2000) Architecting with RM-ODP. Prentice-Hall, Englewood Cliffs. isbn 0-13-019116-7

Romero R, Vallecillo A (2005) Modelling the ODP computational viewpoint with UML 2.0. In: Proceedings of 9th International EDOC Conference, Enschede, The Netherlands, pp 169–180, IEEE

Stabell (1998) Configuring value for competitive advantage: on chains, shops, and networks. Strateg Manage J 19(5):413–437

Steen MWA, Akehurst DH, Doest HWL, Lankhorst MM (2004) Supporting viewpoint-oriented enterprise architecture. In: Proceedings of 8th International EDOC Conference, California, USA, pp 201–211, IEEE

Turowski K (2002) Standardized specification of business components. University of Augsburg. URL: http://www.fachkomponenten.de/ Date of Call: 2003-03-07

Vernadat FB (1996) Enterprise modeling and integration: principles and applications. Chapman & Hall, London. isbn 0-412-60550-3

Wegmann A (2003) On the systemic enterprise architecture methodology (SEAM). In: Proceedings of 5th International Conference on Enterprise Information Systems Angers, France, pp 483–49

Wegmann A, Naumenko A (2001) Conceptual modeling of complex systems using an RM-ODP based Ontology. In: Proceedings of 5th International EDOC Conference, Seattle, USA, pp 200–211, IEEE

Wegmann A, Balabko P, Lê LS, Regev G, Rychkova I (2005a) A method and tool for business-IT alignment in enterprise architecture. In: Proceedings of 17th Conference on Advanced Information Systems Engineering Forum, Porto, Portugal, pp 113–118, FEUP Edições

Wegmann A, Regev G, Loison B (2005b) Business and IT alignment with SEAM. In: Proceedings of 1st International Workshop on Requirements Engineering for Business Need, and IT Alignment, Paris

Zachman JA (1987) A framework for information system architecture. IBM Syst J 26(3):276–292