From Business To IT with SEAM: the J2EE Pet Store Example

Irina Rychkova, Gil Regev, Lam-Son Le, Alain Wegmann
School of Communication and Computer Science,
Ecole Polytechnique Fédérale de Lausanne (EPFL),
1015 Ecublens, Switzerland
{Irina.Rychkova, Gil.Regev, LamSon.Le, Alain.Wegmann}@epfl.ch

Abstract

Business and IT alignment demands clear traceability between the applications to be developed and the business requirements. SEAM is a systemic visual approach for modeling systems, including information systems and organizations. This paper illustrates how we represent the business role of an IT application and its platform-specific realization in SEAM. We use the Java Pet Store sample application as an example.

1. Introduction

Business and IT alignment is gaining in importance as organizations expect to earn larger returns from their IT investments. As a result management expects explicit traceability between the applications to be developed and the business requirements. In this paper we demonstrate how to represent the business role of an IT application as well as its platform-specific realization with SEAM [1], a systemic multi-level modeling method. We illustrate this technique with the Java Pet Store application. The Pet Store is a well-known example proposed as a Java blueprint in 2001. This application illustrates how distributed web-based applications can be developed with the Java J2EE platform [2], [3], [4].

Figure 1 illustrates the Pet Store example: Customers connect to the Pet Store website, manage their account, browse the catalog, update their shopping cart, and place purchase orders. Each order, placed by a customer, is fulfilled by an Order Processing Center (OPC) module - a part of the Pet Store application. Orders can be approved or rejected by a Pet Store administrator (shown as the Admin GUI in Figure 1). When an order is approved and payment is verified with the Credit Card Service, the supplier ships the product to the customer.

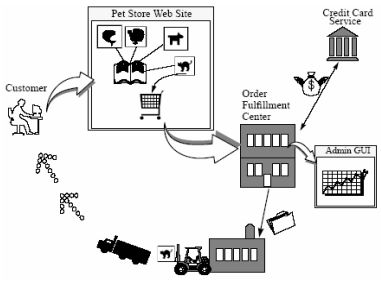

Figure 1. The Pet Store description as represented in the Java BluePrint, [2].

SEAM is a *systemic* and *systematic* modeling method designed to model business and IT systems. A *system* is defined in SEAM as either a configuration of component entities with relationships between them – *system as composite* – or as one entity in which the component entities are abstracted – *system as a whole*. In a system as a whole, the observer can perceive emergent properties that are specific to the whole, but that may not be perceived in the analysis of the parts. In a system as a composite, the observer can visualize the construction of the system. SEAM is a systemic approach because the market segments, the companies, the IT application, the IT modules, and the software components can be represented and modeled as systems. SEAM also makes explicit such system-related concepts as context, lifecycle, and system boundary.

SEAM is a systematic method because we employ the same modeling principles and notations for all systems regardless of their kind. The pictograms might change to reflect the difference in nature of the modeled system (e.g. supply chain style arrow for business entities, cube for IT applications – see Figure 2). However, the specification of the system is done in a same way regardless of the nature of the system. For example, in all systems, properties are

represented with squares and actions with rounded rectangles.

In the Pet Store example, we identify three *organizational levels* that address the structure of the Pet Store starting from its business context to its implementation on the J2EE platform:

- ***the Pet Store segment:*** customer, PetStore administrator, supplier companies, credit card company, and Pet Store application that mediate their interaction;
- ***the Pet Store application***: the Web Site and the OPC modules;
- ***the Web Site module:*** set of J2EE components[1].

These levels are illustrated in Figure 2. The decomposition in three levels is a direct consequence of the Pet Store structure in the original blueprint example.

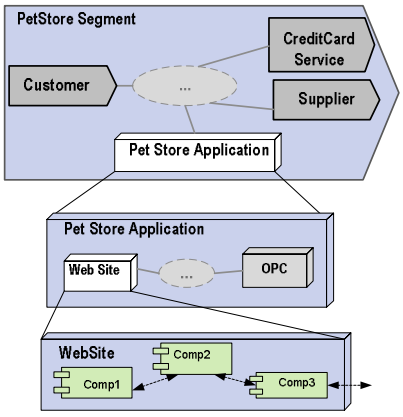

Figure 2. SEAM hierarchical representation of the PetStore example (Figure 1).

In this paper we propose an approach for modeling systems across multiple organizational levels that makes explicit the traceability (i.e. the correspondence) between levels. We use SEAM modeling notation and define two templates for SEAM models. The first template specifies the transition from one organizational level to another when no technological platform is involved; the second template specifies how to integrate the technological platform.

This paper is organized as follows. In Section 2 we present the SEAM hierarchical modeling method and its graphical notation. In Section 3 we model the Pet Store application in its business context and the website module in the Pet Store application. In Section 4 we present how the Pet Store website is realized using J2EE components. Section 5 is an overview of the related work. In Section 6 we present our conclusions.

[1] To respect the space limitations, in this paper we focus on the Pet Store Web Site and omit the OPC.

2. The SEAM Approach for Hierarchical System Modeling

SEAM [1] is a method for modeling general systems, including information systems and organizations. The SEAM epistemological principles are based on General System Thinking (GST) [5] and Living Systems Theory (LST) [6]. The epistemological principles are useful to explain why we perceive reality as hierarchical and how we can relate the different levels in the hierarchy. The SEAM ontology is based on the foundations of the RM-ODP ISO/ITU standard [7]. The ontology defines the concepts used for modeling, such as object, action, activity, state [8].

The SEAM approach defines two hierarchies for its models: the functional and the organizational level hierarchies.

The functional hierarchy is a set of system specifications, in which the system is modeled as a whole whereas its behavior can be modeled as a whole or as a composite (i.e. an action is decomposed into multiple actions).

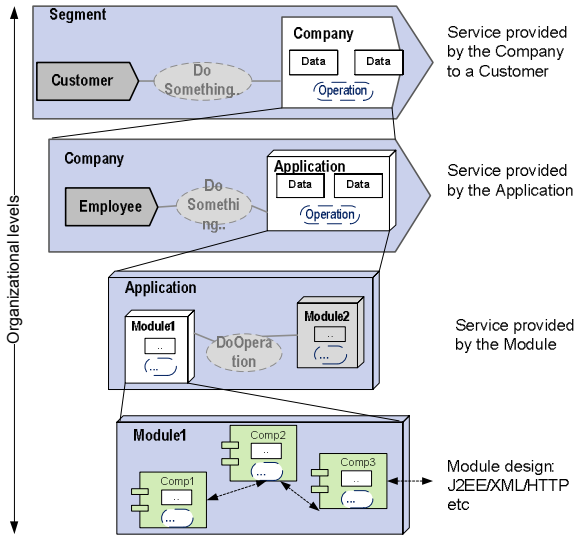

Figure 3. SEAM hierarchical approach to general system modeling.

The organizational hierarchy can be interpreted as a set of system specifications that makes explicit the systems' construction (i.e. a system is decomposed into multiple systems). In general, modelers can specify as many functional and organizational levels as they need. Figure 3 illustrates a typical organizational hierarchy.

A system is represented in the model as a *working object*. A working object can be specified as a. **whole** (black box) (e.g. App1 in Figure 4 (a)) or as a **composite** (white box) (e.g. App1 in Figure 4 (b)).

A specification as a whole describes the working object by its observable *properties* and its *localized actions*. For example, App1 in Figure 4 (a) has Data1 and Data2 as observable properties and _DoX is a localized action that represents the responsibility of the application App1 in the collaboration DoX.

A specification as a composite describes the working object as a set of component working objects participating in a *collaboration*. App1 in Figure 4 (b) is represented as a set of modules Mod1, Mod2, and Mod3 participating in the DoY collaboration. Here the concept of collaboration stands for an action performed by more than one system.

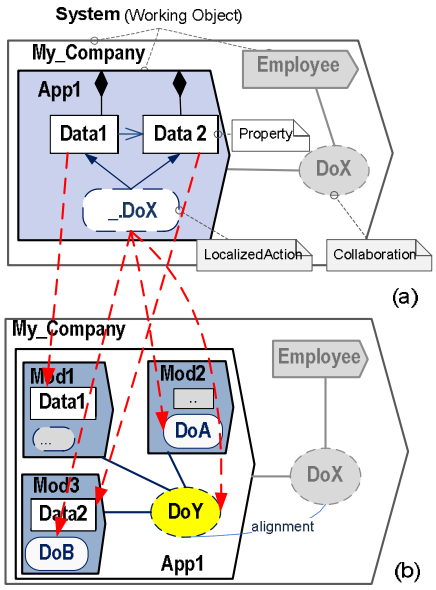

Figure 4. SEAM whole/composite template: from platform-independent to platform-independent specification.

In Figure 4 the specification of App1 as a whole is mapped to the specification of App1 as a composite. The dashed lines show how the properties and the behavior of the application App1 (black box) are distributed between the components of application App1 (white box). For example, Data1 in App1 as a whole is mapped to Data1 in Mod1, component of App1. In a similar manner, action _.DoX is split into actions DoA and DoB and these actions are assigned to modules Mod2 and Mod3, components of App1. The transition from whole to composite also defines behavioral constraints for the specified components. For example, to guarantee that the combination of DoA and DoB is behaviorally equivalent to the localized action _.DoX, a specific constraint is required. These constraints are captured in collaborations (here DoY). We define **alignment** as the behavioral equivalence between a system specification as a whole the specification of the same system as a composite [9].

Figure 5 illustrates the transition from Mod3 as a whole (Figure 5 (a)) to Mod3 as a composite (Figure 5 (c)), in which Mod3 is implemented on a specific technological platform. The X-platform architecture and design patterns are used to specify a generic _Module_ (Figure 5 (b)) as a set of *platform-specific* components operating together. The transformation is done by a mapping of the properties and localized actions (namely, Data2 and DoB) of Mod3 as a whole to the structure, specified by generic _Module_.

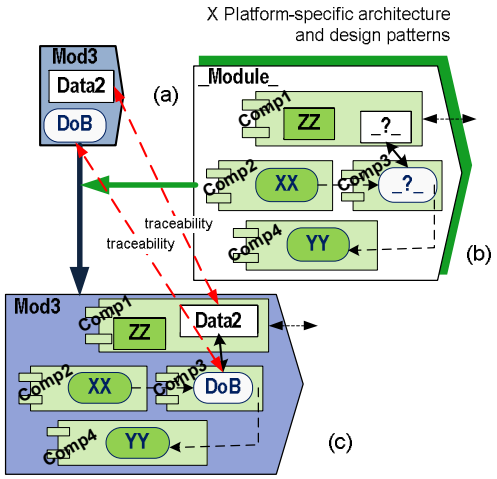

Figure 5. SEAM whole/composite template: from platform-independent to platform specific specification.

Using this approach, **traceability** (or correspondence) between system specifications at different organizational levels is established.

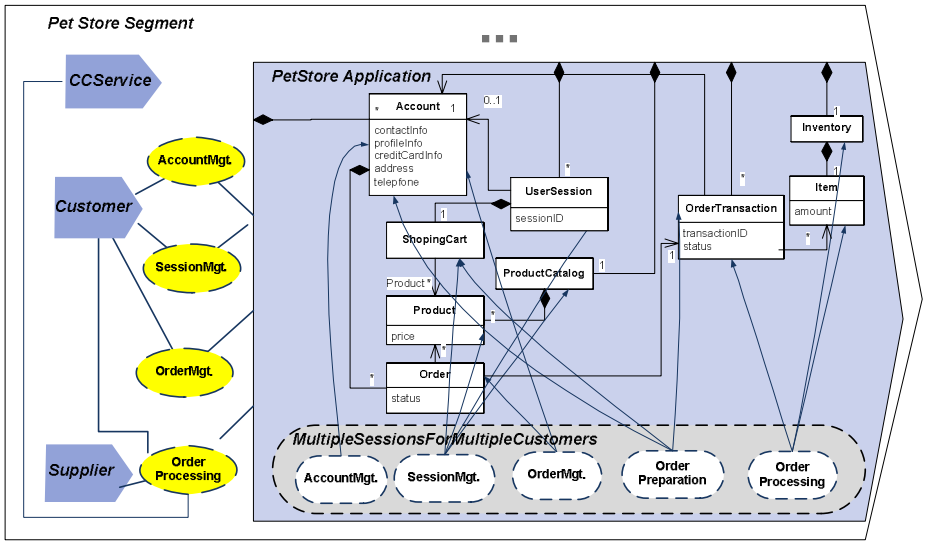

Figure 6. Pet Store Application modeled as a whole.

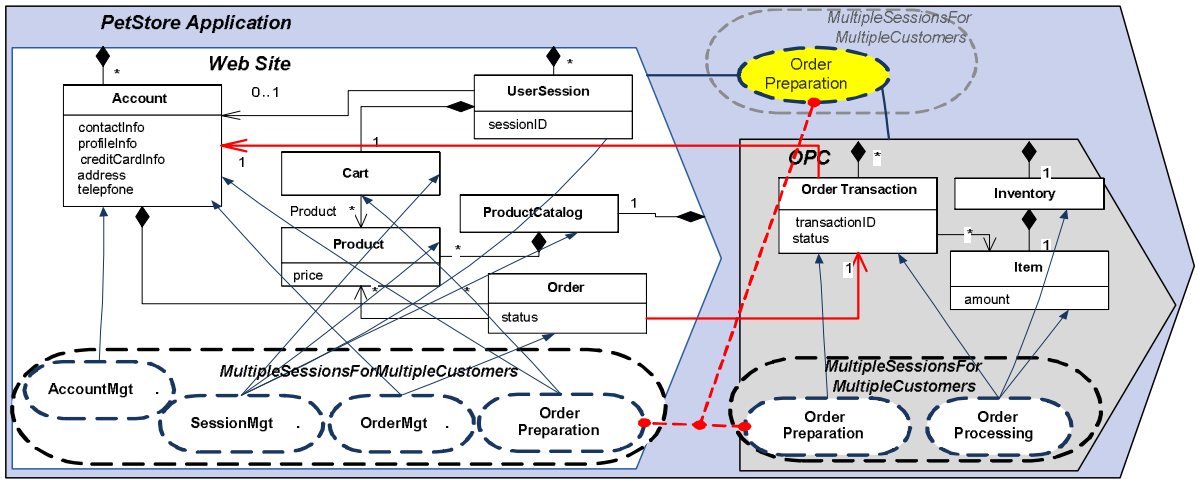

Figure 7. Pet Store Application modeled as a composite; the Web Site and the OPC modules modeled as wholes.

In the next sections we illustrate the SEAM modeling technique on the example of the Pet Store. First, we specify the PetStore application and then we model the Web Site and OPC modules using the template illustrated in Figure 4 Then we present a mapping of this model to the J2EE platform using the template illustrated in Figure 5. Note that this work does not aim to discuss or criticize the J2EE architecture of the Pet Store but to focus on the traceability, defined above.

3. From the Pet Store Application to Pet Store Web Site and OPC Modules.

Based on the Pet Store description illustrated in Figure 1-2 we introduce a model of the Pet Store Application as it is shown in Figure 6. Properties are defined based on the specification in [2]. We define the most general behavior of the PetStore Application: a MultipleSessionForMultipleCustomers. As the sessions can be interleaved, the actions (e.g. AccountMgt, OrderMgt, etc) can appear in any order. This is why no control flow is visible in the specification. SEAM allows the modeler to focus on one session for one

customer (OneSessionForOneCustomer, not shown) in which case the execution constraints become visible.

As described in the Java blueprint, the Pet Store application is composed of a WebSite and an OPC module. Using the template, represented in Figure 4, the modeler can define the PetStore application as a composite (Figure 7) from the specification of the Pet Store as a whole (Figure 6). The OrderPreparation action is distributed between the Web Site and OPC modules. This is why it appears as a localized action in both modules as well as a collaboration in the PetStore application. We define the actions for the **Pet Store Web Site** as follows:

- Account management – to create or delete an account, to update customer details;
- Order management – to browse the product catalog, place, and track customer orders;
- Session management – to provide user authorization (sign in and out) and to maintain a shopping cart;
- Order preparation – to collect and verify the customer and order data;

For the Order Processing Center (OPC) the actions are:

- Order preparation – to receive the order from the website and to initialize the order transaction;
- Order processing – to contact suppliers, approve orders, verify payments, manage the inventory, and organize shipping.

From now on we will focus on the Pet Store website and omit the OPC.

4. Pet Store Web Site: J2EE Platform-Specific Model

The Pet Store website is designed following the Model-View-Controller (MVC) architecture [2] and implemented using J2EE multi-tier model.

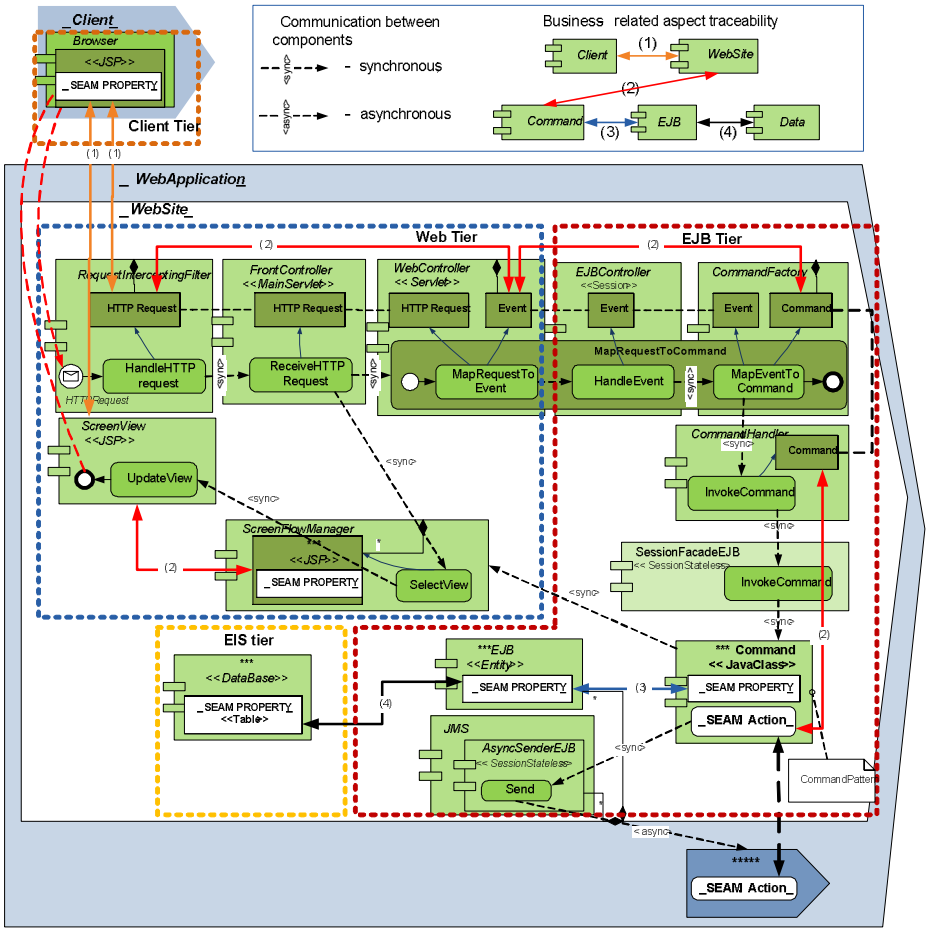

Figure 8. J2EE specification of the MVC architecture pattern for a generic _WebSite_.

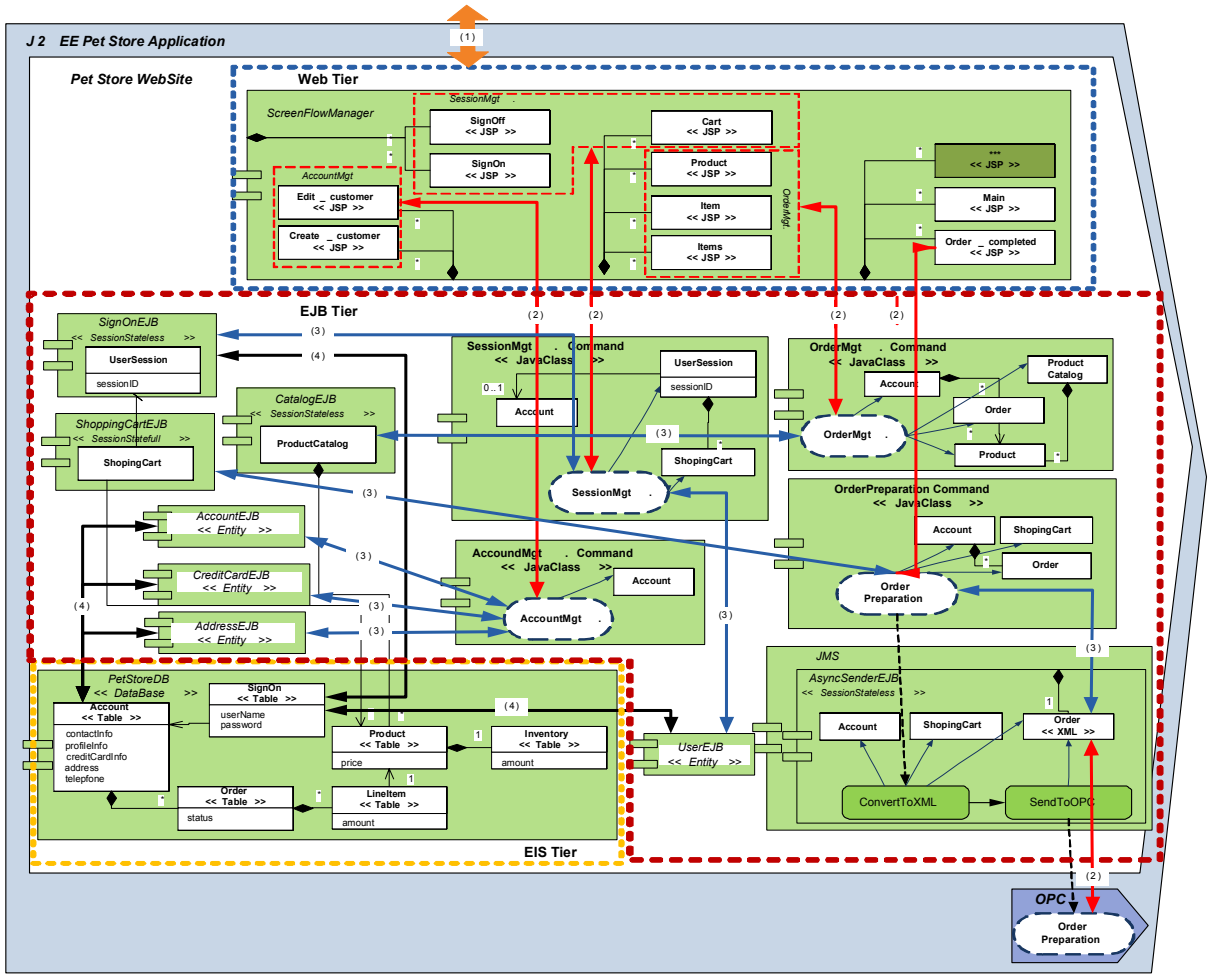

Figure 9. Platform-specific model of the Pet Store Web Site: design using MVC architecture on top of multi-tier J2EE model. Web Site communicates with a customer using an interface (1); customer can invoke a command (2) that is identified with a SEAM action the command operates with a specific Java Bean (3) to access the corresponding data stored in the database (4).

MVC architecture is beneficial for the Pet Store Web Site module, where a user interacts with a web site by multiple request-response iterations.

The J2EE platform provides a multi-tier distributed model for its applications. It defines the following tiers:

- ***A client tier -*** to implement the user interface and the user communication with the server;
- ***A Web tier*** - to implement the client services through Web containers;
- ***An EJB tier -*** to implement the business logic through Enterprise Java Bean (EJB) containers.
- ***A EIS (back-end) tier*** - to implement the persistence through the Enterprise Information Systems (EIS) using standard APIs.

Each tier contains tier-specific components that provide specific features (such as scalability, transactions, security, etc.). The client tier supports a variety of client types, e.g. web browsers (thin clients), Java applets, and Java applications (thick clients). Web tier handles client requests, invokes business logic, and transmits data in response to incoming requests. These functionalities are implemented using Java Server Pages (JSP) and Java Servlets. The EJB tier hosts application-specific business logic and provides system-level services (i.e. data transaction management, concurrency control, security) using Enterprise Java Beans. Three types of beans are defined: Entity Beans, Session Beans, and Message Driven Beans. The EIS tier hosts a database. For more detailed information on the J2EE platform

specification, please see [2]In the previous Section, the SEAM business specification of the Pet Store application and Pet Store website as a whole have been presented. We now illustrate how the J2EE specification of the Pet Store website can be developed.

Using the template, illustrated in Figure 5, the J2EE pattern for generic _WebSite_ is defined (Figure 8). This pattern shows how exactly a website is implemented using J2EE and makes explicit the traceability relations between the specifications illustrated in Figure 7 and Figure 9. For example, the Odrer property of the WebSite as a whole (Figure 7) corresponds to the Order properties of the OrderManagementClass, Order of the OrderPreparationCommand, Order (XML) of JMS and then eventually a table in the PetStoreDB, all visible in the Pet Store Web Site as a composite (Figure 9). Similarly, behavior specified for the PetStore WebSite as a whole is mapped to the PetStore as a composite. For example, OrderMgt (Figure 7) is equivalent to OrderMgt in OrderMgtCommand (Figure 9).

5. Related Work

Model Driven Architecture (MDA) [10] is a design approach that promotes the separation of system functionality specification from its implementation on any technology-specific platform. MDA structures specifications as three different types of models: CIM (computation-independent model), PIM (platform-independent model), and PSM (platform-specific model). The SEAM organizational levels have analogies with these different kinds of models. However, SEAM is based on its own ontology, different from UML. The relation between SEAM and MDA was addressed in more detail in [11].

QVT (Queries/Views/Transformations) [12] is a standard for model transformation in the model-driven architecture (MDA). The abstract syntax of QVT is defined by MOF (Meta-Object Facility). SEAM proposes a relational approach (according to the classification in [16]) for model transformation that targets model alignment [9].

Almeida et. al [17] define a model-driven design trajectory for context-aware services. Their approach defines three modeling levels that differ by their degree of abstraction and platform-independence. Models on the highest level describe a service behavior from the external perspective, abstracting out the environment – that corresponds to the black box specification in SEAM; the lowest level represents a service realization on the concrete platform; the middle level specifies the service realization on the abstract platform called A-MUSE. In this level, service behavior is described from the internal perspective that corresponds to SEAM white box specification. The transformation of a high-level service specification into the middle-level platform-independent service design in [17] requires that the behavior of the assembly of abstract components, defined by A-MUSE, corresponds to the behavior defined by service specification. The alignment of the SEAM models is defined in a similar way.

RM-ODP [7] defines five viewpoints which are based on the ontology defined in Part 2 of the standard. SEAM specifies systems that are also defined using the Part 2 concepts. However, we do not have viewpoints and use organizational and functional levels instead. SEAM models are traceable (or aligned) across hierarchical levels.

Catalysis [18] 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.

KobrA [19] proposes a recursive model that describes IT systems/components. KobrA is based on UML. KobrA differs from SEAM by its tight link to the UML meta-model (as opposed to RM-ODP). Even if this method can model multiple systems, it is designed to focus mainly on one system of interest. J2EE Pet Store is also used as a modeling example to demonstrate the KobrA method [tbd].

6. Conclusions

This paper introduces the SEAM hierarchical method for modeling applications from business requirements to their platform-specific implementation. We define two main modeling templates to relate the specification of a system as a whole and the specification of a system of a composite. One of the templates is applicable when no technological platform is involved. The second template specifies how to integrate the technological platform. The main benefits of SEAM hierarchical method is the traceability between the application to be developed on a concrete platform and its business requirements. This is especially useful for teaching.

Significant work is needed to provide tool support for platform-independent to platform-specific transformation. Currently, we provide tool support for modeling systems as whole and as composite, independently of any technological platform. Further

work also involves developing patterns for different platforms, such as .NET. Examples of the .NET Pet Store can be found in [20], [21].

We believe that this systematic and systemic visual approach can be beneficial for representing and comparing different platforms as well as for reasoning on business and IT alignment.

References

[1] A.Wegmann, "On the systemic enterprise architecture methodology (SEAM)", *Proceedings of International Conference on Enterprise Information Systems (ICEIS) (2003)*

[2] I., Singh, B., Stearns, M., Johnson, and the Enterprise Team: *"Designing Enterprise Applications with the J2EETM Platform"*, Second Edition. Addison-Wesley, (2002).

[3] Java BluePrints, "Sample Application Design and Implementation", http://java.sun.com/blueprints/guidelines

[4] SUN Microsystems, "Java Pet Store Sample application." http://java.sun.com/developer/releases/petstore

[5] G.M.,Weinberg, *"An Introduction to General Systems Thinking",* Wiley & Sons (1975)

[6] J.G., Miller, *"Living Systems"*. University of Colorado Press, (1995)

[7] OMG, *"Reference model of open distributed processing",* Draft International Standard (DIS) (1995)

[8] A., Wegmann, A., Naumenko, "Conceptual Modeling of Complex Systems Using an RM-ODP Based Ontology", *Proceedings of 5-th IEEE International Enterprise Distributed Object Computing Conference (EDOC)* (2001)

[9] A., Wegmann, P., Balabko, L.S., Le, G., Regev, I., Rychkova, "A Method and Tool for Business-IT Alignment in Enterprise Architecture", *Proceedings of CAiSE Forum* (2005)

[10] MG, *"MDA Guide",* Version 1.0.1 (2003) http://www.omg.org/docs/omg/03-06-01.pdf

[11] O., Preiss, A., Wegmann, "MDA in Enterprise Architecture? The Living System Theory to the Rescue…", *Proceedings of 7th IEEE International Enterprise Distributed Object Computing Conference* (EDOC 2003) p.2-13 , 2003

[12] OMG, Meta object facility (MOF) 2.0 Query/View/Transformation Specification. (2005)

[13] C., Atkinson, B., Paech, J., Reinhold, T., Sander, "Developing and applying component-based model-driven architectures in KobrA". *Proceedings of 5th International EDOC Conference*, IEEE. (2001)

[14] *Unified Modeling Language (UML)*, v. 2.1.1. OMG (2007)

[15] D.F., D'souza, A.C., Wills, *"Object, Components and Frameworks with UML, The Catalysis Approach",* Addison-Wesley (1999)

[16] K., Czarnecki, S., Helsen, "Classification of Model Transformation Approaches", *Proceedings of OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture* (2003)

[17] J.P.A., Almeida, M.-E.., Iacob, H., Jonkers, D., Quartel, "Model-Driven Development of Context-Aware Services", *Proceedings of DAIS 2006*, LNCS 4025, pp. 213 – 227, (2006)

[18] D. F., D'souza, A.C., Wills, *Object, Components and Frameworks with UML, The Catalysis Approach*, Addison-Wesley, isbn 0-201-31012-0 (1999)

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

[20] Microsoft, "Using .NET to Implement Sun Microsystems' Java Pet Store J2EE BluePrint Application", Version 2.0, (2002),: http://msdn.microsoft.com/library/

[21] G., Leake, J., Duff, "Microsoft .NET Pet Shop 3.x: Design Patterns and Architecture of the .NET Pet Shop", (2003) http://msdn.microsoft.com/library/