SeamCAD: Object-Oriented Modeling Tool for Hierarchical Systems in Enterprise Architecture

Lam-Son Lê, Alain Wegmann
Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
CH-1015 Lausanne, Switzerland
{LamSon.Le, Alain.Wegmann}@epfl.ch

Abstract

Enterprise Architecture (EA) requires modeling enterprises across multiple levels (from markets down to IT systems). Providing tool support for such models is a challenge (e.g. model containment hierarchy, navigation difficulties, problems to relate elements between different diagrams). In this paper, we identify the requirements that a CAD tool needs to satisfy to manage such hierarchical models. We then propose a solution to meet these requirements: SeamCAD - a tool designed to manage hierarchical models. We present the key features of SeamCAD and an overview of the modeling language it uses. The benefit of the proposed solution is tool support for managing enterprise models.

Keywords: CAD tool, Enterprise Architecture, Service Engineering, Living System Theory, RM-ODP, UML, Requirement Engineering, Business Thinking, Systemic Enterprise Architecture Methodology

1. Introduction

The goal of enterprise architecture is to align the business systems and the IT systems in order to improve enterprise's competitiveness [1]. Enterprise architecture (EA) deals with hierarchical systems that typically span from business entities (e.g. market, company, department…) down to IT components (e.g. applications, applets, servlets, beans…). During an EA project, the EA team – typically a multi-disciplinary team - develops an enterprise model that represents the enterprise and its environment. Working with a model is important; when making the model, the team develops an agreed representation of the enterprise, of its environment and of what the project needs to achieve.

Modeling hierarchical systems is challenging for the modelers and for the Computer Aided Design (CAD) tool designers. In EA, the model represents multiple systems. Each system has its functionality represented at different levels of detail. The modeler can easily get lost while navigating in such a model. In addition, the modeler needs to be able to represent the relationships between the elements shown in these different levels. This is called traceability.

Computer Aided Software Engineering (CASE) tools, that are developed for software development, usually focus only on one system - the system to design. It is thus difficult to use them to design multiple hierarchical systems. For example, quite frequently, these CASE tools provide only one name space and problems appear if a same identifier is used in more than one context. Most of them use the Unified Modeling Language (UML) [2]. UML provides different kinds of diagrams to express various aspects of an IT system (e.g. use-case diagrams for the requirements, class diagrams for the design, activity diagrams for the implementation etc…). Each diagram has specific model elements that can be graphically connected. Putting model elements created in all the diagrams into a single data model with an explicit containment hierarchy, while preserving relations between these elements, is a major challenge. Generic modeling frameworks and domain-specific tools take new approaches and can express the system hierarchy to some degree. Unfortunately, their model navigation and their notation are not designed to visually show the containment hierarchy (e.g. the notation rarely provides nested graphical elements).

In this paper, we define the requirements for a CAD[1] tool for modeling hierarchical systems in EA (Section 2). We then present a solution that fulfills these requirements (Section 3). Our solution, SeamCAD, is a CAD tool specifically designed to model hierarchical systems. Next, we present the related work in CAD tools for hierarchical system design (Section 4). Section 5 draws some conclusion and outlines our future work.

[1] We call our tool a CAD tool because its scope is mainly towards marketing and business process modeling rather than software modeling (even if software modeling is possible). CASE (Computer Aided Software Engineering) tools are used for modeling software.

2. Requirements for a CAD Tool

In this section, we first give an example of an enterprise model and describe some of its important features (Section 2.1). We then present the requirements for a CAD tool managing such an enterprise model (Section 2.2).

We focus on requirements related to the representation of functionality across hierarchical systems. We do not consider other requirements which are not specific to functional modeling of hierarchical systems (e.g. quality attribute or nonfunctional requirements modeling, version management…).

2.1. Example

The example describes a bookstore whose management decides to provide the company's services via the Internet. The management creates an EA team who is in charge of this project. Figure 1 presents a simplified representation of the environment of the enterprise and of its organization. To fully analyze the impact of this project, the EA team has to reason about the multiple levels shown in Figure 1: i.e. the market level, the business system level, the company level, the department level, and the IT level. For each level, we describe the kind of analysis made by the EA team.

The *market level* represents the business systems (i.e. group of companies) active in the market. This level is useful to reason about the overall customer experience, regardless of the companies' responsibilities in providing the services to the customer.

The *business system level* represents companies or individuals that work together to achieve a commercial goal. In our example, the business system of the bookstore (*BookCoBis*) is composed of the book publisher (*PubCo*), of the company itself (*BookCo*), of the shipping company (*ShipCo*) and of the bank. The business system of the customer (*CustomerBis*) is composed of the customer, the bank and the shipping company that delivers the books. This level is useful to reason about the company's responsibilities (e.g. outsourcing strategies in *BookCoBis* or purchasing decision making unit in *CustomerBis*).

The *company level* represents the departments operating inside the company. In our example, *BookCo* has a purchasing department (*PurchasingDep*) that collaborates with the warehouse department (*WarehouseDep*) for the processing of customer orders. This level is useful to reason about business processes and department's responsibilities.

The *department level* represents employees and IT systems. In our example, the purchasing department (*PurchasingDep*) consists of a clerk and of an order processing application (*OpApp*). One of the main goals of the project is to redesign *OpApp*; but the project also needs to redefine the responsibilities of the employees and of the departments. So, this level is useful to specify the business processes together with the IT systems' requirements and the employees' job descriptions.

It is possible to have additional levels for describing the IT system implementation (e.g. *IT level, server level*, *component level* and *Java class level*).

In summary, enterprise models need to describe the enterprise's environment and the enterprise's organization. This is done through a hierarchy of systems (e.g. market, business system, company, department, IT system…).

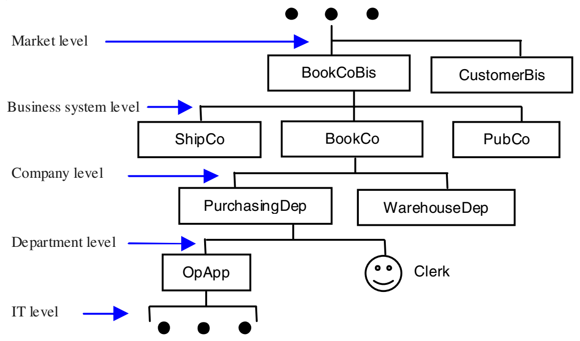

Figure 1. Informal representation of the systems described in the example of the online bookstore

The enterprise models need to capture the functionality of all the relevant systems, at different levels of details. In our example, the EA team needs to express that *BookCo* as a company (at the business system level) performs an action called *Market* in which books are sold. At the company level, the *BookCo's Market* action becomes collaboration between the *BookCo's* departments. Each department has its own responsibility in this collaboration.

In summary, enterprise models need to describe the functionality of all relevant systems (from market down to IT) at different levels of details. Our experience has shown that such models, even if they do not describe all aspects of an enterprise, are useful to define the functional responsibilities of the company, of the departments/employee and of the IT systems. This is especially useful when a company needs to change its strategies. Based on this common functional model, the multiple specialists (e.g. finance, security...) can develop their own models. [3] presents how this kind of model was used in a concrete industrial project.

2.2. Tool Requirements

Let us consider a CAD tool that would manage the enterprise models described in Section 2.1. Figure 2 describes the tool and the people who would use it, the EA team members. The universe of discourse (UoD) represents the perceived reality of the team members. In the UoD, the team members perceive entities. Examples of entities are markets, business systems, or actions performed by them. These entities are represented as model elements in the enterprise model.

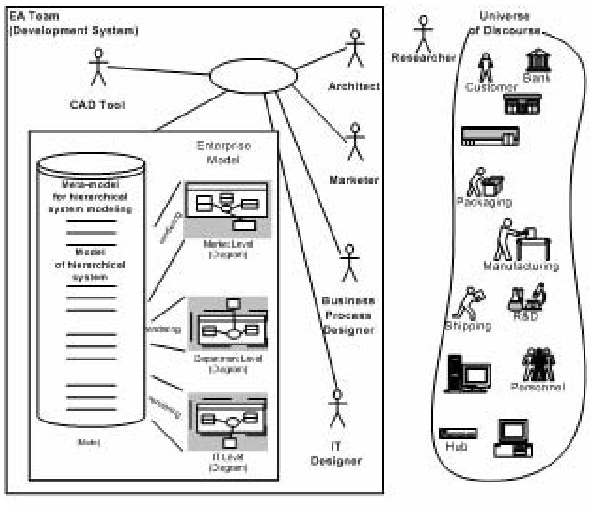

Figure 2. The context of the CAD tool

In the EA team, there are specialists such as marketers, business process designers, and IT designers. They are responsible for managing specific entities. For instance, the marketers reason about business systems and markets. The business process designers manage business processes. All of them use the CAD tool to build the common enterprise model. In general, each specialist is in charge of a specific level in the model. The enterprise architect coordinates the specialists. Her goal is to insure the alignment between all levels. The CAD tool can help her validate this alignment.

The CAD tool should allow the different specialists to work within the same enterprise model at the level for which they are responsible. It is thus essential that the tool shall explicitly manage an *organizational hierarchy* that represents the enterprise's environment and organization. This is the first requirement.

We have seen in Section 2.1 that a system's functionality needs to be modeled at different levels of details that make up the *functional hierarchy*. This is the second requirement. Note that one of the challenges for the tool designer is to provide an ergonomic way to manage these two hierarchies (i.e. enterprise's environment/organization and levels of details in the functionality). If this is not achieved, the modeler might get confused between these two hierarchies.

The members of the EA team expect to reason on graphical representations of the enterprise model. In addition, graphical models are well adapted to represent systems as they make relations between systems more intuitive [4]. This leads to the third requirement that includes the following three characteristics:

- The notation should be systemic. This means that it should be well adapted to represent hierarchical systems. For example, all systems could be modeled in a uniform way regardless of their nature[2]. The notation should also emphasize concepts related to systems like traceability between levels, relations between a system and its environment, containment hierarchy of systems…

- The notation should be discipline-specific, so that the specialists can visually recognize what they are responsible of. Although the systems are represented in a uniform manner (e.g. all systems have properties and participate to actions), the visual elements that represent the different kinds of systems can change between organizational levels. For example, IT people might want to use UML subsystems to represent IT systems. Business people might want to use the Porter arrow rather than UML subsystems to represent companies [6].

- The notation should be close to UML [2] whenever possible, so that UML practitioners can have an intuitive feeling of what the notation represents.

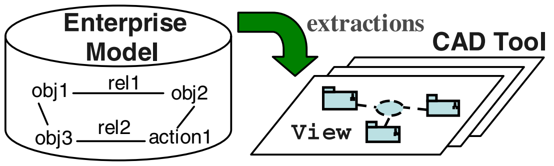

Figure 3. Diagrams are extracted from the model

We also need to specify the way the CAD tool manages the model and the diagrams. In most of the modeling tools, diagrams are normally listed and organized into folders. Quite often, a graphical element such as a class or actor is created in one diagram and will appear in other diagrams. Sometimes, the synchronization between these diagrams creates problems. This synchronization is crucial when modeling hierarchical systems in EA. It is very frequent that elements appear in multiple diagrams. For instance, a company will appear in multiple diagrams. If the name of the company changes, all diagrams need to change. For this reason, diagrams should be generated by extracting the relevant elements and their relationships from a common model

[2] This is one of the key features of a systemic approach. A systemic approach is based on system theory. The Cambridge Dictionary of Philosophy [5] defines "system theory" as the "trans-disciplinary study of the abstract organization of phenomena, independent of their substance, type, or spatial or temporal scale of existence".

(see Figure 3) in a similar way that domain-specific modeling tools such as GME [7] work. In addition, the traceability between model elements shown in different diagrams need to be stored in the common model too. This common model is organized according an ontology designed to support the modeling of hierarchical systems. This is the fourth requirement.

To summarize the requirements, the CAD tool shall:

i. manage an explicit organizational level hierarchy that represents the enterprise's environment and organization;
ii. manage an explicit functional level hierarchy for the systems that represent the systems' functionality at different levels of detail;
iii. have a notation which is systemic, discipline-specific, understandable by UML practitioners;
iv. have a common ontology-like model from which the diagrams are generated

3. Modeling Hierarchical Systems with SeamCAD

Our solution for the requirements defined in Section 2.2 is SeamCAD: a CAD tool for SEAM (Systemic Enterprise Architecture Methodology) that is specifically designed for modeling hierarchical systems. An overview of SEAM is available in [8].

In Section 3.1 we give a short overview of the SEAM modeling language used in SeamCAD. The complete presentation of the modeling language can be found in [9]. In Section 3.2 we present the tool itself.

3.1. SEAM Modeling Language Overview

The key feature of our modeling language is its capability to systematically represent systems at different levels of detail. We call *organizational level hierarchy* the levels in the model that describe the enterprise's environment and its organization. We call *functional level hierarchy* the description of the systems functionality at multiple levels of details. The SEAM modeling language explicitly defines these two hierarchies.

The SEAM modeling language is comparable to approaches such as the Unified Modeling Language (UML) Profile for Business Modeling [2], Catalysis [10], KobrA [11], Systems Modeling Language [12] and Object Process Methodology (OPM) [33]. The main differences between SEAM and these approaches lie in the fact that the SEAM modeling language is systemic (i.e. it represents hierarchical systems across organizational levels and functional levels in a systematic and explicit manner) and can represent multiples systems in details.

In this subsection, we first present the SEAM language definition (Section 3.1.1) followed by the SEAM notation (Section 3.1.2). To make this section more concrete, Figure 4 provides a simplified representation of the enterprise model described in Section 2.1. We use Figure 4 to illustrate the concepts introduced in this section.

3.1.1. SEAM Language Definition. In each organizational level (e.g. market level, business system level, company level, etc...) we represent systems with model elements called computational objects. These computational objects can be seen as whole (i.e. showing their functionality) or as composite (i.e. showing their construction).

Computational objects participate in joint actions. For example, in Figure 4 (a), *BookCo*, *PubCo* and *ShipCo* participate in the joint action *mfg_sale*. For each computational object participating in a joint action, there is a transaction and a localized action. For instance, *BookCo* takes part in *mfg_sale* and has the localized action *Market* and the transaction *MarketTxn*. The localized action represents the local responsibility of the computational object with respect to a given joint action.

The state of the computational object is modified by the joint actions and by the localized actions. The state is captured with information objects that belong to the computational object. Joint actions and localized actions change the state of the information objects (not visible in Figure 4). The transaction is a special kind of information object that exists only when a localized action is performed. The transaction is useful to represent the behavioral context in which information objects exists. For example, within a transaction, it is possible to represent parameters and temporary information objects that exist during the occurrence of the corresponding localized action.

The joint action, the localized action, and the transaction can be viewed either as whole or as composite. The composite view is useful to see the details of the functionality. For example, in Figure 4 (b), the joint action *market*, the localized actions *Market* and the transactions *MarketTxn* (of *WarehouseDep* as well as of *PurchasingDep*) are all seen as whole. They are however composite in Figure 4 (c). This enables the modeler to understand the details of the joint action *market* that is actually refined into three smaller ones: *select*, *order* and *pack*. In both *PurchasingDep* and *WarehouseDep*, she can also see the details of the localized action *Market* and of the transaction *MarketTxn*.

Note that in Figure 4, 3-dot symbols stand for additional organizational levels the modeler may want to express in the enterprise model. They are linked together with Figure 4 (a), (b) and (d) by dashed lines to show the traceability along the organizational level hierarchy.

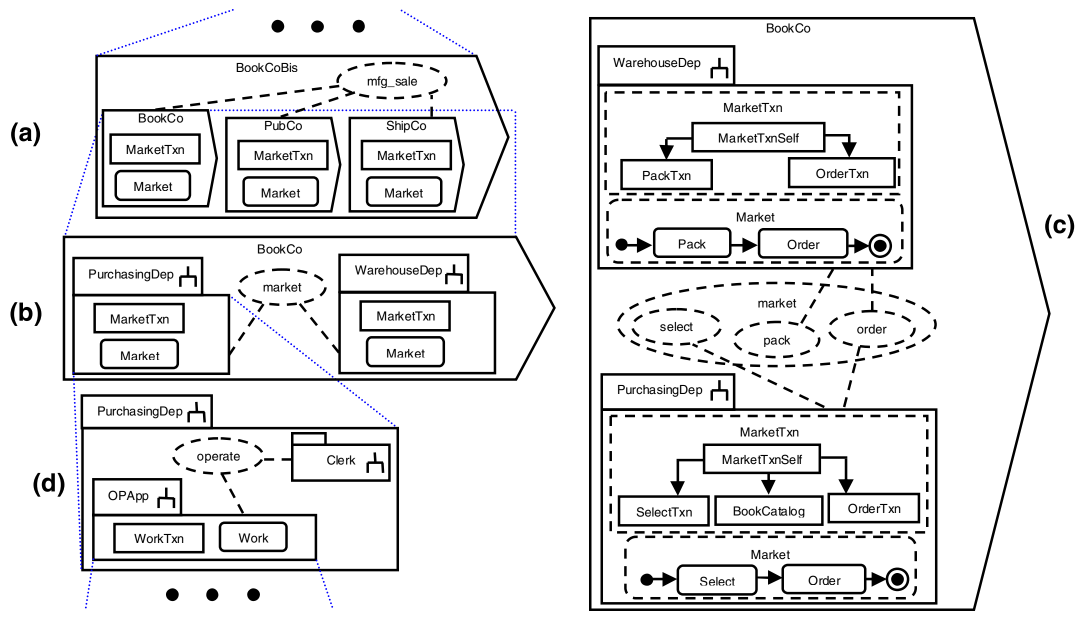

Figure 4. SEAM model of the on-line bookstore: a) Business system organizational level, b) & c) Company organizational level at two functional levels and d) Department organizational level

The SEAM modeling language takes its foundation from Miller's "Theory of Living Systems" (LST) [13] and from the ITU/ISO standard "Reference Model of Open Distributed Computing" (RM-ODP) [14]. In the Living System Theory, the concept of level is used for reasoning about any living system, from individual cells to supranational organizations such as the United Nations Organization. We borrow that concept of level and call it the organizational level. In SEAM, we reason from Java classes (and not from individual cells) to supranational organization (like LST). Within these organizational levels, we use RM-ODP to represent what is perceived in the reality. All the systems in Figure 4 are RM-ODP computational objects. The state of the computational objects is captured by RM-ODP's information objects. The RM-ODP concept of action has been specialized by taking the concepts of "joint action" and "localized action" from Catalysis [10]. In short, the SEAM modeling language has four main concepts: computational objects, information objects, joint actions, and localized actions. All these model elements can be viewed either as whole or as composite. This way of interpretation comes from the concepts of composite and atomicity defined in RM-ODP.

Our synthesis of LST and of RM-ODP was done in the following way: first we formalized the foundations of RM-ODP (basic terms such as objects, actions, etc…) in Alloy - a light-weight specification language based on set theory [15]. The result of this formalization is available in [16]. Then we improved these definitions to add the concepts of organizational levels (from LST) and functional levels [17], which resulted in the SEAM modeling language [9]. SeamCAD is built based on this modeling language.

3.1.2. SEAM Notation. In order to match the different notations used by different specialists within the EA team, the computational objects (i.e. the systems) can have custom graphical pictograms. To represent markets, business systems and companies, the modeler can decide to use Porter arrows [6] (visible in Figure 4 (a) and (b)). The Porter arrow takes its origins in the representation of the value systems and the value chains that were made popular by Porter. For departments, IT systems and software components, the modeler can decide to use the UML subsystem graphical element (visible in Figure 4 (b) and (d)). In future versions of the tool, additional pictograms for computational objects will be added to reflect other systems that may appear in enterprise models.

The SEAM modeling language has 4 main concepts: computational object, information object, joint action and localized action. We already discussed how the computational objects are represented. The representation of the other three model elements is borrowed from UML:

- The SEAM joint action (dashed ellipses in Figure 4) represents the participation to an action of a set of computational objects. It can be represented by the UML collaboration.

- The SEAM localized action (rounded rectangles in Figure 4) represents the behavior of a particular computational object. So it can be drawn using UML "action state" - the graphical element used in activity diagrams.

- The information objects (rectangles in Figure 4), that represent properties of computational objects, are drawn using the UML classes (without operations).

As discussed in Section 2.2, one of the challenges is to visually show model containment in diagrams. This can be achieved by nesting our graphical elements, which are actually 2D pictograms (e.g. rectangle, ellipse…). Dashed lines are used to render composite elements (with the exception of the joint action which is always dashed to be closer to UML). In this way, nesting can be made to as many levels as the modeler wants in the diagram.

The SEAM modeling language takes graphical elements from UML whenever possible, so that practitioners can recognize the elements. However, SEAM is a research language that we target for hierarchical modeling. For this reason, our meta-model and notation are optimized for modeling hierarchical systems. This is one of our main originalities compared to UML 1.x [2]. Graphical elements of UML 1.x are rarely designed to be nested; the composition is, in general, represented via a composite aggregation relation. Domain-specific modeling tools like GME [7] or MetaEdit++ [18] do not support visual containment neither. In UML 2.0 the situation has improved. For instance, the Superstructure Specification extends the meta-class "Class" with the capability to have internal structure and ports [2]. For a general comparison between SEAM and UML, the reader can refer to [19] that presents how the UML meta-model could be simplified if it was based on RM-ODP (as SEAM does).

3.2. Key Features of SeamCAD

We show in this section the way SeamCAD[3] addresses the requirements identified in Section 2.2. The tool features are explained using the example introduced in Section 2.1 and illustrated in Figure 4.

3.2.1. Explicit Hierarchy that Represents the Enterprise's Environment and Organization. SeamCAD allows the modeler to work on multiple organizational levels. The tool has a main window that shows the organizational level hierarchy of the model in a tree view. This main window enables the user to open modeling windows in which specific parts of the model can diagrammatically be edited.

Figure 5 shows a modeling window. The tree view of a modeling window looks the same as the one in the main window. The modeler can interact with it to generate the diagram she wants to display. Selecting a computational object in the tree node will generate a diagram on the right panel of the modeling window. For example, in Figure 5, the tree node of *BookCoBis* is selected. The modeler can see in both the tree view and in the diagram that, at the business system level, *BookCoBis* consists of *BookCo* (purchasing and management), *PubCo* (providing books) and *ShipCo* (delivering books) that collaborate together. Note that Figure 5 corresponds to Figure 4 (a).

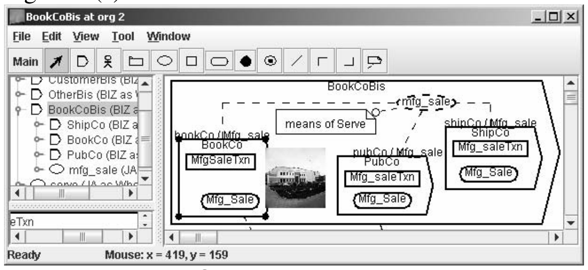

Figure 5: *BookCoBis* and its companies seen as wholes (business system organizational level)

Each modeling window is dedicated to a particular organizational level. There is no limit on the number of modeling windows opened at the same time. For instance, the modeler can open 3 modeling windows and select *BookCoBis*, *BookCo* and *PurchasingDep* to see the organizational levels described in the example. The tool ensures the consistency among all modeling windows. Changes made in a window will propagate to the others.

In the tree view, the top node represents the model (*BookCoProject* in our example). Its only child node stands for the first computational object of the organizational hierarchy. Below this node, the organizational level hierarchy is visible as described in Section 2.1.

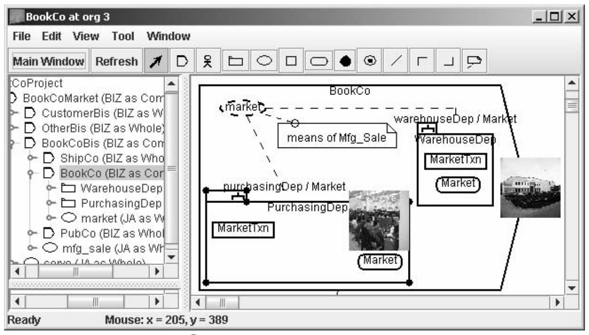

Figure 6. *BookCo* and its departments at company organizational level, functional level 1.

The most frequent user interactions are selecting and expanding/collapsing a tree node. If the modeler expands a computational object, it is equivalent to changing to a subsequent organizational level. For example, the modeler expands and selects the tree node of *BookCo* in

[3] SeamCAD is available at http://seamcad.epfl.ch.

Figure 5, which is at the business system level. SeamCAD then displays a window shown in Figure 6 expressing the departmental structure of *BookCo* - the company level. At this level, there are two departments: *PurchasingDep* responsible for IT management of customer orders and books, *WarehouseDep* responsible for inventory processing and packaging. In Figure 6, the modeler has chosen to hide *BookCo*'s environment. In contrast, collapsing a computational object is equivalent to changing to a precedent organizational level.

Figure 5 and 6 illustrate what we mean by traceability. The localized action *Mfg_Sale* in *BookCo* as a whole (in Figure 5) is realized by the joint action *market* in *BookCo* as composite (in Figure 6). In Figure 6, the role of *PuchasingDep* is *purchasingDep / Market*. This role is also visible as the *Market* localized action and the *MarketTxn* transaction in *PurchasingDep*. All these relationships are kept in the common model managed automatically by the tool. The user only needs to enter once the name of the joint action (*market*), the name of the system (*PurchasingDep*), the name of the localized action (*Market*) and visually setting *Market*'s means to *market*. The tool will manage the generation of the note (*means of Mfg_Sale*) and role names (*purchasingDep / Market*).

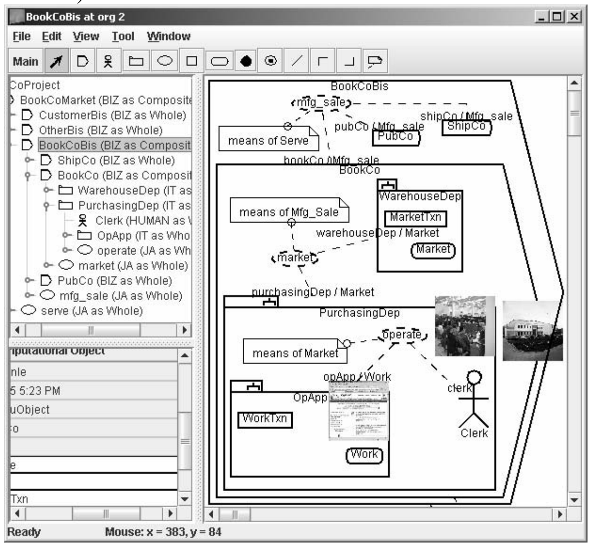

Figure 7. Multi-level representation covering *BookCoBis*, *BookCo*, *PurchasingDep* and *OpApp*.

It is possible to see multiple organizational levels in one window. Figure 7 shows a modeling window in which the business system level, the company level and the department level are represented in one diagram. The user can obtain such a diagram by expanding, in the tree view of Figure 5, the tree nodes *BookCo* and *PurchasingDep*. Figure 7 is a combination of Figure 4 (a), (b) and (d).

3.2.2. Explicit Functional Level Hierarchy. Navigating through the functional level hierarchy without confusing the modeler is a challenge. Two preliminary versions of SeamCAD were developed until we found adequate solutions to this challenge. These preliminary versions of the SeamCAD implemented organizational level and functional level as completely separate concepts. For each computational object, the user could select the functional levels and organizational levels she wanted to display. This lead to problems as it was possible to see diagrams with multiple objects shown at different levels of functionality.

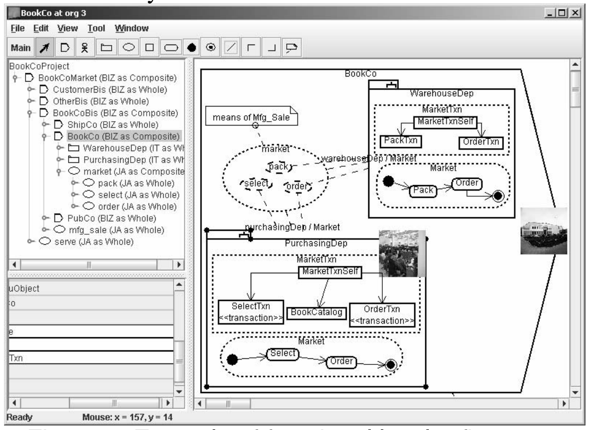

Figure 8. Example of functional level refinement: same organizational level and systems as Figure 6 but behaviors described at functional level 2.

We found a solution by enforcing a given level of functionality in all objects shown in the diagram. All computational objects participating in this joint action are displayed at the same level of functionality. This is achieved by giving to the modeler the choice to view the joint actions as whole or as composite. This feature considerably simplifies the navigation in the functional levels as the concept of functional level is hidden in the notion of joint action as whole or as composite. It also keeps separate the navigation through functional levels (done by selecting how joint actions are represented) from the navigation through organizational levels (done by selecting how computational objects are represented).

Figure 6 and Figure 8 illustrate this point. They correspond to Figure 4 (b) and (c), respectively. The *market* joint action is seen as a composite making the *select*, *order* and *pack* component joint actions visible. So the information viewpoints of *PurchasingDep* and *WarehouseDep* are seen as composite. Note that these joint actions have their equivalence in the participating computational objects. For example, the *select* joint action becomes the *Select* localized action that represents the responsibility of *PurchasingDep*. In this purchasing department, there is a special information object called

SelectTxn that represents the transaction corresponding to the localized action. It is useful when the modeler needs to model information objects related to the execution of the *Select* localized action. There is also an information object called *BookCatalog* representing the list of books available to the customer from the perspective of *PurchasingDep*. SeamCAD ensures the synchronization between the joint actions, the localized actions and the transactions of the objects participating in the joint action.

3.2.3. Notation which is Systemic, Discipline-Specific, Understandable by UML Practitioners. As presented in Section 3.1.2, SeamCAD uses discipline-specific graphical elements to represent the computational objects. For instance, in Figure 7, a Porter arrow represents the company and the UML subsystem represents the departments. To make the notation even more concrete, the modeler can attach pictures to a computational object. For example, in Figure 7, the plant picture is associated with the *BookCo* model element, or the cubicle picture with *PurchasingDep*. This feature helps the modeler to recognize what she is looking at.

The systemic notation has multiple features. The exhaustive presentation of these features is out of the scope of this paper. We can still mention few of them:

- Explicit context representation: model elements such as localized actions, information objects and joint actions are always represented within a computational object. The computational object makes explicit the system in which these elements are defined. In a similar way, component actions are always represented within the composite action that contains them. This makes the behavioral context in which actions are explicitly defined. For instance, in Figure 8, the *Select*, *Order* and *Pack* actions are within *Market* making it visible that they define what *Market* means. Note that this feature is implemented thanks to our visually nested notation. The tool automatically resizes an enclosing pictogram whenever the modeler moves nested ones.

- Multiple system representations: As multiple computational objects are shown in a same diagram, the modeler can look simultaneously at the specification of multiple systems at the same time. For instance, in Figure 8, it is possible to analyze the behavior of both *PurchasingDep* and *WarehouseDep*.

- Holistic representation of state and behavior: In a computational object seen as whole, information objects and localized actions can both be visible. Work currently under way investigates how to graphically represent pre and post conditions between the localized actions and the information objects. Thanks to this, the diagram can fully describe actions without requiring additional information such as Object Constraint Language code.

The SeamCAD notation is strongly inspired by UML. Many graphical elements come from UML. The main differences are that SeamCAD notation permits putting all kinds of model element in any diagrams and that SEAM graphical elements are designed to be nested to visually show the containment hierarchy. Note that UML 2 does propose Composite Structure but with only one nesting level [2].

3.2.4. Common Model from which the Diagrams are Generated. SeamCAD has a common model that is outlined in the tree view of multiple modeling windows and of the main window. The modeler has filtering options to control the diagram generation. Three ways of filtering exists:

- It is possible to filter out a specific computational object or joint action. The element that is filtered out is hidden in all diagrams. In the tree view, the corresponding tree node is grayed. The synchronization between diagrams will make this element disappear in all diagrams. For instance, *WarehouseDep* is hidden and its tree node is grayed in Figure 9.

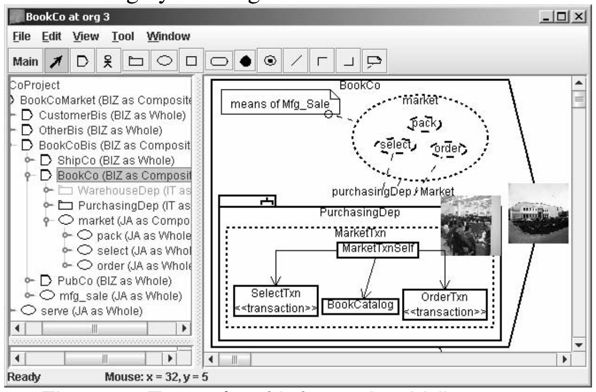

Figure 9. Example of information hiding: same organizational level and functional level as Figure 8 but *PurchasingDep*'s behavior and the entire *WarehouseDep* computational object are hidden.

- The modeler can decide to hide the environment of a computational object. This is done when the modeling window is created. For example, Figure 7 makes *BookCo*'s environment visible whereas Figure 8 hides it. Making the environment visible is a powerful feature as it allows the modeler to make the knowledge of the system of its environment explicit. SeamCAD enables drawing a "trace" relationship between an information object in a computational object and a model element in the computational object's environment to represent the fact that there is a relation between a system and its environment.

- A last feature allows the modeler to filter out the information objects or localized actions, or both, of a particular computational object. With this feature, the modeler can obtain diagrams that are close to UML diagrams. For example, in Figure 9, it is possible to see a UML class diagram inside the *PurchasingDep* computational object by hiding its actions. This capability illustrates that SeamCAD could be considered as a UML-like tool in which the context can be systematically represented.

The challenge of maintaining the traceability between elements shown in different diagrams (Section 3.2.1) and the synchronization within diagrams (Section 3.2.2) can be resolved by carefully designing relations in the common ontology-like model.

4. Related Work

Today, there exist quite a large number of object-oriented modeling tools and generic modeling frameworks. The object-oriented modeling tools can be roughly categorized into two main groups: software modeling and enterprise modeling. The former aims at providing UML diagrams and some functions to automate the development process (e.g. reverse engineering, code generation, report generation…). The latter provides the modeler with some extra diagrams (may not be UML-compatible) for modeling business processes, organizational units, etc... There are also generic modeling frameworks that allow modelers to quickly define a domain-specific modeling tool.

Rational Software [20], Visual UML [21], UML Studio [22], UML Suite [23], Poseidon [24], Objecteering UML Modeler [25], Object Domain [26], Microsoft Visio with UML template [27] etc… can be considered as software modeling tools. They support a wide range of UML diagrams that are generally organized into folders or views. These folders and views are typically originated from UML taxonomy on diagrams such as static structure, use-case, implementation etc. This taxonomy is unfortunately not suitable for the representation of the hierarchy of organizational and functional levels. To model a hierarchical system with these tools, the modeler builds several diagrams with the assumption that each of them corresponds to an organizational level. As a consequence, the modeler sees neither the hierarchy of the organizational level nor the traceability between diagrams. In short, we find that the modeler cannot effectively navigate her hierarchical models with these tools.

Enterprise Architect [28], System Architect [29], Mega [30], Arc Styler [31], etc… can be considered as enterprise modeling tools. They either provide extra modeling diagrams (beyond UML) or allow the modeler to customize UML diagrams. For example, with Enterprise Architect it is possible to draw any UML element in a specific diagram. The modeler can use UML collaborations, UML actors and UML classes to represent business systems and people collaborating together. However, these tools are still diagram-based. The same comments about model navigation which we made about software modeling tools also apply to enterprise modeling tools.

OpCat [32], the tool for OPM [33], is more suitable for modeling hierarchical systems because it supports zoom-in/zoom-out operations. In addition, OpCat is a model-based tool. Its diagrams can be created on-demand when the user zooms-in to a process or an object. However, since OpCat does not natively address hierarchical systems, its navigation panel is not used for browsing the hierarchy. It lists diagrams instead.

MetaEdit+ [18] is considered as a generic modeling tool. The basic rationale behind MetaEdit+ is, at the meta-level, most of modeling tools essentially defines different kinds of objects having some properties and relationships between them. Its main advantage is the ability to quickly define a tool for a given modeling language. Nevertheless, in the aspect as a generalized diagram-based modeling tools, MetaEdit+ also shares the shortcomings with software modeling tools regarding hierarchical systems analyzed above.

GEF [34] allows developers to create a graphical editor for an existing application model. This framework can be used on top of EMF [34], another framework for data storage, to build a particular modeling tool for hierarchical systems. The main drawback is that the tool built in this way can only be executed within Eclipse and apparently requires quite heavy programming burden. Additionally, the tool graphical pictogram must depend on 2D engineering of GEF, which does not natively support nested notation.

GME is a configurable tool suite that facilitates domain-specific modeling [7]. In GME meta-model, the concept Model can contain other Models, allowing the modeler to establish containment hierarchy in her project. We notice that the tree-view navigation and the way of generating modeling diagrams in SeamCAD are similar to those in GME. The main difference lies in the fact that our tool specifically addresses hierarchical systems in EA by having two model containment hierarchies (functional and organizational) whereas GME was motivated from control systems and integrated circuits (notation is not nested, lack of collaboration modeling).

5. Conclusion

In this paper we presented the requirements for a CAD tool for enterprise architecture. Considering enterprises as hierarchical systems, we analyzed what requirements EA CAD tools need to satisfy the modeling of such systems. We identified four needs: 1. the capability to manage organizational level hierarchy (to represent systems' context and construction); 2. the capability to manage the functional level hierarchy (to describe the systems behavior at multiple levels of details); 3. the use of a systemic, discipline-specific notation, understandable by UML practitioners; 4. the use of a common model from which diagrams are extracted.

The paper then presents an implementation of these requirements: SeamCAD. It is a tool specifically designed to manage hierarchies. Its originalities lie in the navigation tree that presents the organizational hierarchy and in the capability to see joint actions between systems as wholes or as composite (which hides the complexity of the functional hierarchies) as well as the nested notation. Thanks to the theoretical foundations of the supported modeling language, SeamCAD can provide complete traceability between the diagrams. It manages a central model that can be viewed and edited through multiple modeling windows. Special care has been taken on selecting the graphical notation in order to help the modeler to recognize in which level she works while keeping a uniform way of reasoning about all levels (as all levels use the same systemic meta-model to define the model elements).

SeamCAD has been used in one project in industry involving business process reengineering and in a 2-year construction project for a new building for our school. In this last project, an enterprise model of the school was made. This model was useful to specify how the building should be equipped and what IT system should be installed in the building.

Future work includes the definition of an operational semantics for the SEAM language [35]. With these semantics we will be able to perform model checking (e.g. to compare two representations of a same system, represented in two different organizational levels or at two different functional levels) and model simulation. Additional research and development directions include the development of course material for marketing, requirement engineering and enterprise architecture based on SeamCAD.

References

[1] Schekkerman, J., *How to Survive in the Jungle of Enterprise Architecture Framework: Creating or Choosing an Enterprise Architecture Framework*: Trafford, 2004.
[2] OMG, Unified Modeling Language, http://www.uml.org/
[3] Wegmann, A., Regev, G., and Loison, B., "Business and IT Alignment with SEAM," presented at REBNITA / 13th IEEE RE workshop, Paris, September 2005.
[4] Durand, D., *Que sais-je? La Systémique*. Paris: Presses Universitaires de France, 1979.
[5] Audi, R., *The Cambridge Dictionary of Philosophy*: Cambridge University Press, 1999.
[6] Porter, M. E., *Competitive Advantage*: Free Press, 1985.
[7] Karsai, G., Maroti, M., Ledeczi, A., Gray, J., and Sztipanovits, J., "Composition and cloning in modeling and meta-modeling," *IEEE Transactions on Control Systems Technology*, vol. 12, pp. 263-278.
[8] Wegmann, A., "On the Systemic Enterprise Architecture Methodology (SEAM)," presented at 5th ICEIS, Angers, France, April 2003.
[9] Lê, L. S. and Wegmann, A., "Definition of an Object-Oriented Modeling Language for Enterprise Architecture," presented at 38th Hawaii International Conference on System Sciences, Hawaii, USA, January 2005.
[10] D'souza, D. F. and Wills, A. C., *Object, Components and Frameworks with UML, The Catalysis Approach*: Addison-Wesley, 1999.
[11] Atkinson, C., Paech, B., Reinhold, J., and Sander, T., "Developing and applying component-based model-driven architectures in KobrA," presented at 5th IEEE EDOC, Seattle, USA, September 2001.
[12] Systems Modeling Language (SysML), http://www.sysml.org/
[13] Miller, J. G., *Living Systems*: University of Colorado Press, 1995.
[14] OMG, "ISO/IEC 10746-1, 2, 3, 4 | ITU-T Recommendation, X.901, X.902, X.903, X.904, Reference Model of Open Distributed Processing," 1995-1996.
[15] MIT, The Alloy Constraint Analyzer, http://alloy.mit.edu/
[16] Naumenko, A., Wegmann, A., Genilloud, G., and Frank, W. F., "Proposal for a formal foundation of RM-ODP concepts," presented at WOODPECKER / 3rd ICEIS workshop, Setúbal, Portugal, July 2001.
[17] Lê, L. S. and Wegmann, A., "An RM-ODP Based Ontology and a CAD Tool for Modeling Hierarchical Systems in Enterprise Architecture," presented at WODPEC / 9th EDOC workshop, Enschede, The Netherlands, September 2005.
[18] MetaCase, MetaEdit+, http://www.metacase.com
[19] Naumenko, A. and Wegmann, A., "A Metamodel for the Unified Modeling Language," presented at <<UML>> 2002, Dresden, Germany, September/October 2002.
[20] IBM Rational Software, http://www-306.ibm.com/software/rational/
[21] Visual UML, http://www.visualobject.com/
[22] UML Studio, http://www.pragsoft.com/
[23] UML Suite, http://www.telelogic.com/
[24] Poseidon, http://www.gentleware.com/
[25] Objecteering UML Modeler, http://www.objecteering.com/
[26] Object Domain, http://www.objectdomain.com/
[27] Microsoft, Microsoft Visio, http://www.microsoft.com
[28] Enterprise Architect, http://www.sparxsystems.com.au
[29] System Architect, http://www.popkin.com/
[30] Mega, http://www.mega.com/
[31] Arc Styler, http://www.io-software.com/
[32] Dori, D., Reinhartz-Beger, I., and Sturm, A., "OPCAT - A Bimodal CASE Tool for Object-Process Based System Development," presented at 5th ICEIS, Angers, France, April 2003.
[33] Dori, D., *Object-Process Methodology, A Holistic Systems Paradigm*: Springer Verlag, 2002.
[34] Eclipse Platform, http://www.eclipse.org
[35] Wegmann, A., Balabko, P., Lê, L. S., Regev, G., and Rychkova, I., "A Method and Tool for Business-IT Alignment in Enterprise Architecture," presented at 17th CAiSE Forum, Porto, Portugal, June 2005.