ON THE SYSTEMIC ENTERPRISE ARCHITECTURE METHODOLOGY (SEAM)

Prof. Alain Wegmann
Systemic Modeling Laboratory (LAMS)
School of Computer and Communication Sciences (IC)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
Email: alain.wegmann@epfl.ch
WWW: http://lamswww.epfl.ch

Keywords: enterprise architecture, system sciences, business engineering, software engineering.

Abstract: This paper presents an original methodology for Enterprise Architecture. Enterprise Architecture is the discipline whose purpose is to align more effectively the strategies of enterprises together with their processes and their resources (business and IT). Enterprise architecture is complex because it involves different types of practitioners with different goals and practices. Enterprise Architecture can be seen as an art; it is largely based on experience but does not have strong theoretical foundations. As a consequence, it is difficult to teach, to apply, and to support with computer-aided tools. This paper presents how system sciences, by defining the concept of the systemic paradigm, can provide these necessary theoretical foundations. Thanks to our systemic paradigm, the enterprise architects can improve their understanding of the existing methodologies and thus find explanations for the practical problems they encounter. This paper then gives a concrete example of the application of the systemic paradigm: the Systemic Enterprise Architecture Methodology (SEAM) - an original methodology. With SEAM, architects can use a methodology that alleviates most of their practical problems and that can be supported by a tool.

1. INTRODUCTION

Business and information technology (IT) integration is essential for enterprises to achieve their competitiveness. Unfortunately, a large number of the IT projects fail in achieving this integration (Standish Group, 1994). Enterprise Architecture (EA) addresses this issue and goes beyond IT. EA addresses business implementation in general and, more specifically, the integration in efficient business processes of the IT resources (e.g. applications, clusters, networks …) and of the business resources (e.g. facilities, people, machines …). Unfortunately, according to the analysts' forecasts, the success rate of the EA projects is not much higher than that of the IT projects (MetaGroup, 2001).

The EA methodologies are largely based on the experience developed by the architects across the multiple projects they have realized. The experience and the good practices are captured by means of patterns that are reused from project to project. Beyond these patterns, the methodologies have no theoretical foundations. This leads to challenges in promoting/ teaching/ applying EA and to the difficulty of developing efficient tools to support EA. In other words, EA has not reached the level of maturity that it deserves in regards to its importance to the development of competitive enterprises.

Our group's goal is to bring more maturity to EA. To do so, we analyze the foundations of EA and formalize them in a systemic paradigm. Then, based on this paradigm, we develop the *Systemic Enterprise Architecture Methodology (SEAM).* The SEAM acronym also refers to the *seamless* integration between business and IT. SEAM

Published at the International Conference on Enterprise Information Systems 2003 (ICEIS 2003), Angers, France. Electronic version available at: http://lamswww.epfl.ch.

includes the *SEAM philosophy,* the *SEAM method,* and prototypes of computer-aided design (CAD) tools. The SEAM philosophy corresponds to the fundamental principles on which SEAM is built. These principles are important as they define the formal models required to develop the CAD tools necessary to support EA.

This paper has the following structure: Section 2 - presents what EA is and its main challenges; Section 3 - what SEAM is; Section 4 - a small case study.

2. ENTERPRISE ARCHITECTURE AND ITS CHALLENGES

In this section, we present EA in general. We then introduce two existing methodologies. We complete our presentation by a discussion of the EA challenges.

An *enterprise* is an organization of resources, which performs a process. Examples of resources are people, computers, machines, buildings … *Architecture* is the "manner in which the elements are arranged or organized" (Merriam-Webster, Web). So, EA is the discipline that deals with the organization of the enterprise's resources. This organization evolves, as a consequence of the forces that are exerted in and on the enterprise. The goal of an EA project is to define and implement the strategies that will guide the enterprise in its evolution. These strategies are actually both the plans to be realized by the enterprise and the patterns stating how the enterprise operates (Mintzberg & al., 1998). To make this more concrete, let's take an example of an EA project related to an on-line bookstore (BookCo). Our example is inspired by a New-York Times article describing Amazon, the on-line retailer (Hansell, 2001). As with Amazon, forces are exerted on BookCo to require its profitability. As a consequence, the management team investigates how to reduce the BookCo operating and capital expenditures while maintaining the same revenue. To do this, different possible organizations of the BookCo resources are analyzed. The solution that is selected consists in using the warehousing resource of the publisher instead of the one of BookCo. By choosing this strategy, BookCo focuses on the sales and marketing and leaves the logistic aspects to the publisher. This would reduce the operating and capital expenditures and allow BookCo to become profitable. As a consequence, this strategy requires the modification of the existing business processes, of the IT applications, and the renegotiation of the contracts with the publisher.

EA projects deals with the enterprise in all its aspects. As a consequence, EA teams have to be multi-disciplinary. An EA team includes specialists (typically upper management, functional managers and senior staff members) together with architects. The role of the architect is to federate the efforts of the specialists to ensure successful projects. The architects are either from the enterprise itself or from a consulting firm or an IT vendor. In our example, the EA team would be composed of the management team of BookCo (marketing, operation, IT, finance, legal, and quality managers), the senior staff members (senior business analysts and IT developers) and one or more enterprise architects.

In an EA project, the EA team analyzes the existing organizations and designs new organizations. To reason and communicate about these organizations, the team develops an enterprise model. An *enterprise model* represents the resources found in the enterprise and in its environment, together with the processes in which they participate. The model represents only the entities of the enterprise and of its environment that are relevant for the project. The enterprise model is structured in organizational levels (Miller, 1995). In EA, an *organizational level* is a part of the enterprise model that describes the enterprise from the viewpoint of one or more specialists. Traditionally EA methodologies consider three organizational levels. The *business level* represents the company, in its market. It is generally analyzed in terms of products or services, and revenue. The *operation level* represents that the company is composed of people and systems (e.g. warehouse or IT application). The operation level is generally analyzed in terms of business processes and operating expenditure. The *technology level* represents the technical infrastructure composing the systems (e.g. machinery in the warehouse or software components in the IT application). The technology level is generally analyzed in terms of capability and capital expenditure. Each level describes either what currently exists (*as-is*) or what should exist (*to-be*). What is actually represented in each organizational level depends on the chosen methodology. In general, it is related to the transport/ storage/ processing of either matter/ energy or information. It is important to highlight that these organizational levels are related. For example, an IT system can be modeled in the operation level as an IT application providing a service and in the technology level as a set of software components that implements the service defined in the operational level. Similarly, a warehouse can be modeled as a service in the

operation level and as people and machinery in the technology level. The *traceability* is the capability to make explicit the relations between related model elements found in the various levels of the enterprise model. As the purpose of EA is to integrate business and IT, this concept of traceability is essential as it allows the EA team to make explicit how the integration between the levels is realized. Typically, the business level defines the goals to be reached and the operational and technology levels show how these goals will be reached. Explicitly establishing relationships between organizational levels is what makes EA projects original compared to other multi-disciplinary projects. In regular multi-disciplinary projects, the specialists use their own models. The traceability between them is far more difficult to establish. The consequence is that it is much more difficult, or even impossible, to check whether the project leads to an integrated solution.

EA methods define the development activities in an EA project. A project begins with the decision of an enterprise to react to or to anticipate a change. The team's first activity consists in modeling the entities, from the enterprise and from its environment, that are relevant to the project. In our example, the team models the business level as-is representing BookCo as unprofitable. They also define the operation level as-is representing BookCo's existing business processes and the related operating expenditures. The expected reaction to the change, or the goal, is modeled as an organizational level to-be. In our example, this corresponds to the business level to-be that models BookCo as profitable. Here a gap is created because the operation level as-is (i.e. BookCo using its own warehouse) does not correspond to what is defined in the business level to-be (i.e. BookCo being profitable). The *gap* is the difference between what exists and what should exist to achieve the goal. To close the gap, the team must develop organizational levels to-be and deploy them. An EA project deals with multiple gaps, typically one per level. Of course, all these gaps correspond to a same enterprise analyzed at different levels; so the resolutions of the different gaps cannot be independent of each other. This is why the EA team needs to find adequate tradeoffs across all levels (as opposed to finding "THE" optimal solution). In our example, the EA team closes the operational level's gap by modeling that BookCo should use the publisher's warehouse to reduce the operating and capital expenditures. The definition of this operation level to-be creates a new gap, at this point in the technology level. This gap states that the IT application does not provide the adequate services for outsourcing the warehouse. This second gap can be resolved either by buying a new application or by developing one (based on what already exists). Deploying these operation levels will involve development expenditures, new operating and capital expenditures. The expenditures from all levels have to be considered in the selection of the adequate tradeoff. It is possible that the development expenditures out-weigh the project's benefits and force its redefinition. In summary, in EA, finding the right tradeoff consists in choosing, within each level, a solution that is feasible, practical and that contributes to the overall goal of the enterprise.

Existing EA methodologies are usually presented in two parts: a framework and a method. Usually, the frameworks are quite sophisticated and the methods are rather simple. The *frameworks*, a term widely used in EA, provide guidelines on how to make the enterprise model. As most EA methodologies are proprietary, we present here two commercial methods: Zachman and CSAM.

The Zachman framework is the first EA framework published (Zachman, 1987). Zachman puts an emphasis on describing what exists on each level of an enterprise. In the simplest version of the framework, Zachman proposes to describe within each level: what things are involved (data); how things are done (function), where things are done (network). The Zachman framework uses an add-hoc notation. No specific CAD tool support is available.

CSAM is the Compaq Services Architecture Methodology (CSAM, 2001). It is a methodology very complementary to Zachman as it focuses on documenting the interlocking web of goals, principles, rationales, obstacles, principles underlying the design and not so much on describing what exists in each level as does Zachman. In CSAM the team analyzes within each level: the goals to achieve, the principles guiding what needs to be done, the underlying rationales, the implication and the obstacles related to what needs to be done. CSAM recommend using, whenever possible, discipline-specific theories. These theories also correspond to the best practices and patterns already existing in each discipline and in EA. For example, CSAM recommends the use of Porter's value chain and value system [Porter] when analyzing the business level. The CSAM framework uses text, as notation, in spreadsheet, as tool.

To conclude this Section, we present three important problems we have identified in the practice of EA. Firstly, despite the fact that the EA frameworks are defined to provide consistent representations of the different organizational levels, it is difficult to clearly establish and maintain the traceability between the levels. As a consequence, these frameworks are difficult to teach and apply.

Secondly, there is no tool to support the use of these frameworks. So no help can be provided to the team for developing the enterprise model and, more importantly, for reusing, validating and maintaining the model. As a consequence, the development of multi-level model is tedious and discouraging. Thirdly, the EA frameworks are not object-oriented and are difficult to relate to UML (Unified Modeling Language) (OMG, web). UML is a graphical object-oriented modeling notation widely used by software engineers and has begun to be adopted by the business engineers. As a consequence, these specialists do not feel comfortable using EA frameworks. These three problems hinder the promotion and the applicability of EA. Ultimately all these problems explain why EA is not very popular, despite its obvious importance.

3. SYSTEMIC PARADIGM & SEAM

In this section, we propose theoretical foundations for EA and illustrate how these foundations can be used in a concrete EA methodology: SEAM.

A *system* is a set of interacting components. Based on Section 2, an enterprise is an organization of resources that performs a process. Hence an enterprise is a system in which the components are the enterprise's resources. Thus, we can anchor EA on system sciences, the discipline that provides the necessary theoretical foundations to model and design systems.

Using system sciences, we can classify systems in two categories: complicated systems and complex systems. *Complicated systems* are systems for which the behavior can be predicted by analyzing the components' interactions. Complicated systems are deterministic systems. Typically a computer is a complicated system. *Complex systems* are systems for which the behavior cannot be predicted by such analysis. Complex systems are non-deterministic. Typically a system including humans, such as a company, is a complex system. It is the co-existence and interaction of complex systems with complicated systems that is the challenge for EA. Architects and specialists are well trained for dealing with complicated systems but are usually far less comfortable with complex systems.

System sciences teach us that, to deal with complex systems, we need to change our way of thinking compared to the one we have when dealing with man-made, artificial systems - or complicated systems. Kühn calls a *paradigm* the set of values, or principles, that we use when we think (Kühn, 1962). Kühn claimed that science evolves through paradigm shifts (a radical change of values) as opposed to evolution (incremental changes of values). We claim that, to address the problems of EA, the architects and the specialists need to make a paradigm shift from the mechanistic paradigm used to understand complicated systems to the systemic paradigm used to understand complex systems. The mechanistic paradigm corresponds to the principles intuitively and implicitly used by most professionals. To work with systems in general, system scientists have shown that professionals need to adopt the systemic paradigm (Lemoigne, 1994). The *systemic paradigm* makes explicit the principles used to reason about systems and proposes a way to structure the disciplines that deal with systems in general (see "system inquiry" in (Banathy, web)). The systemic paradigm defines the concepts of systemic philosophy, systemic/ discipline-specific theories, and systemic method. The *systemic philosophy* explains the concepts used to make models of systems and the relation between these models and the reality. The *systemic/ discipline-specific theories* provide the conceptual tools for teams to reason while working on the model. The *systemic method* explains how to proceed in the analysis and design of systems.

To make explicit the existence of the systemic paradigm is already an important contribution to the field of EA as this provides a theoretical justification for what the parts of the EA methodologies are. For example, in Section 2, Zachman and CSAM provide part of the systemic philosophy by defining the framework. Only CSAM proposes explicitly the use of discipline-specific theories (e.g. Porter's value system). Both methodologies propose a method.

We now present the *SEAM,* our implementation of the systemic paradigm in the field of EA. Section 3.1 presents the SEAM philosophy; Section 3.2 the SEAM method. The theories are not presented as SEAM relies on the theories already existing in the current disciplines involved in EA.

3.1 SEAM Philosophy

The systemic philosophy is composed of three parts that are (Schwarz, 2001): (1) the epistemology defining "what is knowledge" (Section 3.1.1); (2) the ontology defining "what exists" (Section 3.1.2); and (3) the ethics defining "what is right or correct" (Section 3.1.3). Note that most of the discussion on the philosophy is generic and can possibly be applied to most EA methodologies. Only the ontology we use is SEAM specific.

3.1.1 Epistemology

Epistemology is the study of the nature of knowledge and justification [Audi, 1999]. Epistemology defines epistemological principles that are useful for understanding the relationship between reality and the model (Lemoigne, 1994; Checkland, 1999). One of the most important principles is the *constructivism principle*: it states that all knowledge is relative to the observer. As the only way to comprehend reality is to have knowledge about this reality (hence to depend on an observer), observer-independent descriptions of reality do not exist. This principle is fundamental as, among other things, it provides the justification behind the organizational levels found in EA. The concept of level corresponds to the different abstractions, or viewpoints, that the specialists have developed to simplify their understanding of systems. It happens that these viewpoints appear hierarchical and this is why we call them levels (Miller, 1995). Note that these levels correspond to those identified in the most recent software engineering processes (Atkinson, 2001; D'Souza ,1999). Each discipline considers *levels of reality* that are specific sets of entities perceived in reality, entities that the specialists "control" or realize. For example, software engineers realize software components. So software engineers perceive the existence of a "component" reality level. These levels of reality are represented in the model as organizational levels. The entities in the levels of reality are represented as model elements in the organizational levels. Each specialist usually "owns" one organizational level and factors in the other specialists' organizational levels.

Concretely, understanding this constructivism principle allows SEAM to explain the rationale behind the existing EA methodologies and thus allows for more flexibility. For example, our experience shows that, in many cases, it is useful to go beyond the traditional 3 levels used in EA methodologies. In a project that aimed at reengineering the IT infrastructure of a nation-wide department store, with the goal of being able to change prices nation-wide on a daily basis, we identified 12 levels of reality that we represented in 5 organizational levels (from the marketing analysis of what the enterprise expects from the price fixing process, via the enterprise/ region/ store price adaptation process down to the Java classes, stored in EJB components in the cash register infrastructure existing in the stores).

To conclude, we emphasize that the explicit definition of epistemological principles is necessary to set the bases on which the SEAM ontology (presented in the next Section) is built.

3.1.2 SEAM Ontology

In computer sciences, an *ontology* defines a set of concepts and their inter-relations. Note that in philosophy, ontology is synonymous to metaphysics and refers to what exists in reality. In SEAM, we take the computer science's definition; the ontology corresponds to what exists in the model. We leave to metaphysicians the discussion of what truly exists in reality.

Our general system ontology defines the set of concepts and inter-relations necessary to model systems in general. Our ontology (Naumenko, 2002) is based on RM-ODP, an ISO/ITU standard (ISO/IEC 1996). To be able to build a CAD tool, we have formalized RM-ODP in a specification language called Alloy (Jackson 2000). In our ontology, we consider that the model elements are defined by two characteristics: the basic modeling characteristic, and the specification characteristic. In its simplest form, the ontology defines 5 basic modeling characteristics (BMC) - object, action, state, location in time, location in space - and 2 specification characteristics (SC) - type and instance. Model elements are defined by combining a BMC with an SC. For example, to model an exchange of money against some goods, we use a model element with the basic modeling characteristic "action" and with the specification characteristic "type <sale>". The BMC "action" states that the model element represents something happening in reality. The SC "type <sale>" states that there is a predicate named <sale> that further characterizes the "action" model element. This predicate defines the action's pre-condition as "the buyer has money and the seller has goods" and the post-condition as "the buyer has goods and the seller money". All this together defines the model element with the combined characteristics of being an "action type <sale>". This model element refers to all the happenings, existing in the perceived reality, in which goods are exchanged against money. For a more thorough discussion on behavior modeling and behavior representation, see (Balabko, 2002).

Thanks to our object-oriented ontology, we can develop a CAD tool that can support the modeling of enterprises by using a UML-like notation. The existence of this ontology brings a concrete solution to the three general EA problems identified in the conclusion of Section 2.

3.1.3 Ethics

SEAM defines the relationships between the perceived reality and the model (Section 3.1.1) and what kind of model elements are in the model

(Section 3.1.2). These definitions apply for all SEAM projects. But, in a concrete project, an actual enterprise model needs to be developed. The model is the result of the analysis of the perceived reality. Different perceptions are possible and the team will have to choose. For example, an EA team can perceive an enterprise as serving its customer (by selling products) or as serving its shareholder (by raising the share value). The team has to choose if they want to consider the company as selling products to the customer and then consider rising the share value as a constraint to satisfy or if they want to choose the opposite. Depending on the choice, the enterprise model will be different and this will influence the selection of what will be implemented. The *ethics* correspond to the choices that the specialists make when they decide on how they want to model their perceived reality. These choices cannot have any formal justification. The only justification is that the specialists believe that they are right. This is where experience intervenes.

There are benefits to keep ethics as an explicit concept. This allows SEAM to capture where the skills of the team's members intervene. This provides a means to distinguish between what is formal and what is subjective. Last but not least, this also captures the fundamental business and social values of the enterprise; values that will influence the project goal.

3.2 SEAM Method

An enterprise is a complex system as it involves people, autonomous entities, and because it interacts with other complex systems (e.g. customers, competitors, suppliers). The key characteristic of complex systems is continuous evolution. So, the context in which the project is run continuously evolves. This is the reason a SEAM project is iterative. So the specialists can adapt the model to represent the changes that are happening within the enterprise. This also allows the specialists to test and validate with real people in the enterprise the hypothesis made in the model.

SEAM iterations have 3 kinds of development activities: multi-level modeling, multi-level design, and multi-level deployment. These activities might happen sequentially or in parallel.

The goal of the *multi-level modeling* is to make a new model, or to modify an existing model of the enterprise. It is important that the team members agree regularly on what organizational levels are used. The specialists define or modify the corresponding organizational levels in the model. By doing so they agree on what they perceive as existing in terms of goals, processes, and infrastructure.

The goal of the *multi-level design* is to identify gaps and to resolve them as explained in Section 2. By doing so, the team defines what new process and resources need to be developed and deployed.

The goal of the *multi-level deployment* is to transform what is described in each organizational level to-be in artifacts that can be understood (by people or computers). The artifacts might be plans (e.g. for opening a new plant or for the negotiation of a contract) or might be directly executable (e.g. job descriptions or programs). Note that even if artifacts are developed and deployed, it does not necessarily mean that what was developed will be used in practice (Markus, 1994). Enterprises are complex systems; the people, being autonomous, might not have the motivation to use what was developed. For this reason, in SEAM, these motivational issues are considered explicitly in the enterprise model.

4. CASE STUDY

This Section is based on the BookCo example already presented in Section 2 and makes reference to Fig. 1. Our goal in this section is not to show in detail how SEAM works, but rather to give the reader a feel for SEAM's benefits.

4.1 Multi-level Modeling

In the BookCo project, the team defines 4 organizational levels: business level, company level, operation level, and technology level.

Let's consider the company level first, as it is the most relevant to the management team. The company level as-is represents: the BookCo company (BookCo), the publisher (PubCo) and the shipping company (ShipCo) acting together to manufacture and sell (Mfg&Sale) products to the customer. BookCo is not profitable. This is represented by a property of BookCo. For a discussion on property modeling, please refer to (Preiss, 2002). To express the project goal, the company level to-be is defined. It looks the same as the as-is with the only difference that BookCo is profitable. The team then represents the operation level as-is because that organizational level represents the entities that the team wants to work with. They represent BookCo Purchasing, Warehousing, and the IT application (IT) with PubCo and ShipCo acting together to Market a product.

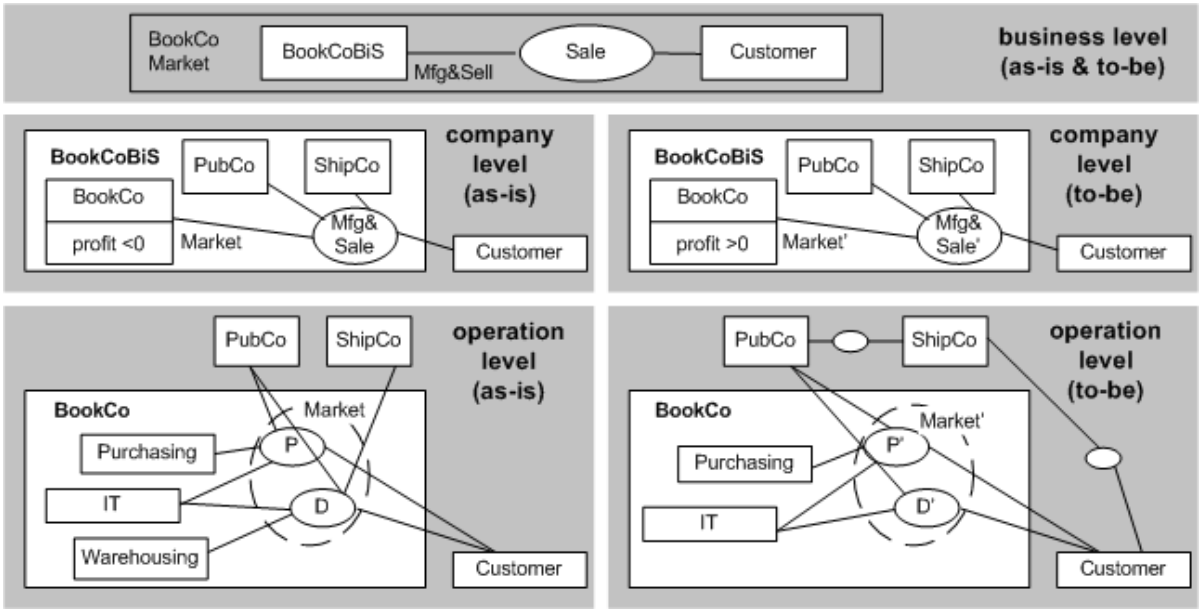

Figure 1: BookCo Project's Organizational Levels As-is & To-Be

Note that the Market action on the operation level corresponds to the role Market done by BookCo in the Mfg&Sale action represented in the company level. The Market action is itself composed of two (component) actions P (for purchasing) and D (for delivery). These are examples of traceability. Traceability is one of the benefits of the use of the ontology as defined in Section 3.1.2.

To completely define the project goal, the team needs to determine what should be maintained (Regev, 2002) between the as-is and the to-be. For this, the team defines the business level. As we mentioned, BookCo and its partners manufacture and sell products. For the customers, it is irrelevant who does what, as long as the customers can get products conveniently. To express this, the team models the BookCo Business System (BookCoBiS) and the Customer. BookCoBiS represents all companies working with BookCo to manufacture and sell products. Note that on the business level, BookCoBiS is considered as a whole and on the operation level as a set of companies. This again illustrates the traceability between levels and the use of our ontology.

4.2 Multi-level Design

The benefit of our design approach is that, for the design of each level (e.g. the operation level), the specialists think both in abstract terms or *goals* (e.g. what is defined on the business level and on the company level) and in concrete terms or *means* (e.g. what is defined on the operation level). This favors the development of better solutions as the specialists can investigate different possible means to satisfy the goal (Hammer, 1990). This is one of the advantages of SEAM.

After having modeled the enterprise across levels, the team then closes the gap found in the operation level by making the operation level to-be. This is done by imagining and analyzing different possible operational levels to-be and selecting the adequate one. The selected solution (as presented in Section 2 and shown in Fig. 1) consists in not involving BookCo in the storage and shipping of the products.

To check the feasibility of the solution the EA team then analyzes and resolves the gap that will exist in the technology level (not represented): the existing IT application does not support the new business process. Working in the technology level is similar to working on the operation level. The only difference between these levels is the use of different discipline-specific theories to assess the various design alternatives.

4.3 Multi-Level Deployment

Multi-level deployment happens as described in Section 3. The project is iterative. In the first iterations, most work will be done in the business-related levels. At these levels, the deployment consists mostly in informing and directing the people about the enterprise's goals (thus possibly

triggering multiple projects to investigate how to implement the goals). Once the business-related levels become more stable, the team will add more technology-related levels. In IT, these technology levels correspond to different IT technologies found in the enterprise such as the application clusters, software components, programming language objects, etc.... Ultimately the IT models will represent the computers, configuration descriptors, programs, etc… that will be physically deployed in the enterprise.

5. CONCLUSION

In this paper, we first present Enterprise Architecture, its importance for developing competitive enterprises and the limitations in its applicability. We then present how a systemic paradigm can provide the theoretical foundations to underlie Enterprise Architecture. To make this more concrete, we present SEAM that extends the EA methodologies, mainly by its use of the philosophy. This explicit use of philosophy allows for the combination of the formal aspects (i.e. ontology) together with the human aspects (i.e. epistemology and ethics). The originality of SEAM is this combination. It enables the integration of a generic approach (from level to level) together with the relevant level-specific theories and practices. It also enables the development of CAD tools that truly support EA. Finally, in our experience, presenting the SEAM philosophy greatly simplifies teaching EA.

ACKNOWLEDGMENTS

I wish to acknowledge the significant contributions made by the SEAM team members: P. Balabko, A. Naumenko, O. Preiss (ABB), G. Regev; our industrial partners: F. Bouchet (consultant), J. Donaldson and L. Laverdure (Compaq Professional Services), G. Genilloud and William Frank (Domain & Financial Systems Architects); our academic partners: E. Schwarz (Uni. of Neuchâtel), and all the other academics who influenced this work.

REFERENCES

Audi R. 1999. The Cambridge Dictionary of Philosophy. *Cambridge Press.*

Atkinson C. & al., 2001. Developing and Applying Component-Based Model-Driven Architectures in Kobra. *IEEE Enterprise Distributed Object Computing Conference.*

Banathy B., web. *A Taste of Systemics*. *www.isss.org*.

Balabko P., Wegmann A. 2002. From the RM-ODP to the Formal Behavior Representation. *In Practical Foundations of Business and Systems Specification. KluwerAcademic Press.*

Checkland P., 1999. Systems Thinking, Systems Practice. *Wiley*.

Compaq, 2002. Compaq Services Architecture Methodology. www.compaq.com/services/arch

D'Souza D. & Wills A., 1999. Objects, Components, and Frameworks with UML. The Catalysis Approach. *Addison Wesley. http://www.catalysis.org*

Hammer M., 1990. Reengineering work: Don't Automate, Obliterate. *Harvard Business Review, 7.*

Hansell S., 2001 May 20, A Front-Row Seat as Amazon Gets Serious. *New-York Times, http://www.nyt.com*

ISO/IEC 1996. X.902, Open Distributed Processing 1995-1996. *http://isotc.iso.ch ('iso/iec standard' in 'ITTF')*

Jackson D. 2000. Alloy: A Lightweight Object Modeling Notation. *http://sdg.lcs.mit.edu/~dnj/*

Kühn T.S., 1962. The Structure of Scientific Revolutions, *Chicago.*

Lemoigne, J.L., 1994. Le constructivisme. *ESF Editeur.*

Markus M. L., Keil M. 1994 Summer. If we Build It, They Will Come: Designing Information System That People Want to Use. *Sloan Management Review.*

Merriam-Webster, Web. M-W Dictionary. www.m-w.com

Metagroup, 2001. The Changing Role of IT Strategy: Enterprise Architecture Strategies. *www.metagroup.com*

Miller J.G., 1995. Living Systems. University of Colorado Press

Mintzberg & al., 1998. Strategy Safari. *Free Press.*

Naumenko A., Wegmann A. , 2002. A Metamodel for the Unified Modeling Language. *UML 2002 Conference.*

OMG, web. Unified Modeling Language. www.omg.org

Preiss O., Wegmann A. 2002. A Systems Perspective on the Quality Description of Software Components. *6th conference on Systemics, Cybernetics and Informatics.*

Porter M.E., 1985. Competitive Advantage. *Free Press.*

Regev G., Wegmann A., 2002. Regulation-Based Linking of Strategic Goals to Business Processes. *Workshop on Goal-Oriented Business Process Modeling.*

Schwarz E., 2001. Personal communication.

Standish Group, 1994. *Chaos Report*. *www.pm2go.com/sample_research/chaos_1994_1.php*

Zachman J.A., 1987. A Framework for Information Systems Architecture. *IBM System Journal, 26, 3.*