Specification and Implementation of a Meta-model for Information Systems Cartography

Gorica Tapandjieva and Alain Wegmann

École Polytechnique Fédérale de Lausanne,
Systemic Modeling Laboratory LAMS
Station 14, CH-1015 Lausanne, Switzerland
`{gorica.tapandjieva,alain.wegmann}@epfl.ch`,
WWW home page: `http://lams.epfl.ch/`

Abstract. Models are used in software engineering, enterprise architecture, requirements engineering, etc. In this context, models can give support in creating a shared understanding of what exists in an enterprise. For this purpose we suggest to use a cartography tool that stores and displays the variety of models. But cartography tools have limitations in the number of enterprise levels shown. Since every model has to conform to a unique meta-model, in this paper we report on a development of a SEAM meta-model used within the Solu-QIQ cartography tool to overcome the limitations of the default cartography meta-models.

Key words: meta-model, SEAM, URBA, enterprise architecture, EA tool, Solu-QIQ

1 Introduction

Enterprise architecture (EA) is defined as "a coherent whole of principles, methods, and models that are used in the design and realization of an enterprise's organizational structure, business processes, information systems, and infrastructure" [8]. Today there are many EA frameworks and methodologies that differ in their approaches and in their level of details. EA comparison studies can assist architects in choosing the appropriate framework or methodology [12, 15]. In this paper we look at EA diagrams as an information systems (IS) cartography.

Having a cartography in an enterprise is important because it helps enterprise architects, managers, governance bodies and all employees in general, to create a shared understanding of what exists within and around the enterprise. With this, decisions that optimize the usage of Information Technology (IT) resources are made more easily. In the literature, an information system cartography is also known as a top-down urbanization approach (IS Urbanization and Enterprise Architecture, URBA-EA) [10], and it comes with the possibility to use a cartography tool.

Any EA tool which outputs a complete EA landscape and which offers a fast way of EA modeling without losing the models correctness can be seen as cartography tool. We decouple the tool from the approach implemented in the tool. As a consequence, in Section 2, we give an overview of few EA tools and EA approaches separately. Section 2 also contains matrices comparing the tools and approaches according to features we find important.

In Section 3 we describe the chosen approach and its meta-model. In Section 4 we show how this meta-model is implemented in the chosen tool. Finally, we give conclusions and present the future work.

2 EA Tools and Approaches

People creating and using models need a modeling tool that speeds up their work. At the same time, their models must not lose the correctness. When we were looking for a tool to do the modeling, we focused on the following criteria:

A. Ability for meta-model customization, making the tool decoupled from it's integrated EA approach.
B. Mass-modeling capability and various formats of data and model import/export.
C. Automatic diagram layout generation.

Our evaluation of some tools is presented in Table 1.

	A	B	C
Iteraplan [6]	No	Yes	Yes
Enterprise Architect [4]	No	Partial	No
Solu-QIQ [3]	Yes	Yes	Yes

Table 1. Summary of some tools evaluation criteria

As for tools, we had several criteria for the EA modeling approach. We looked at frameworks, modeling languages and methodologies, and the questions we asked are:

I. Does the approach have a notation?
II. How many levels (domains) does the approach model?
III. Is the approach's meta-model generic between different levels or domains?
IV. Is the approach declarative?

Our review of some approaches is given in Table 2.

Based on our criteria, we choose Solu-QIQ tool with the SEAM approach. With SEAM's ability to model multiple levels, we model the enterprise from the business down to IT, using the same notation. Also, the declarative way of modeling is an advantage when focusing on macro scenario descriptions. Solu-QIQ makes this approach more powerful with it's mass modeling and automatic diagram-generation features.

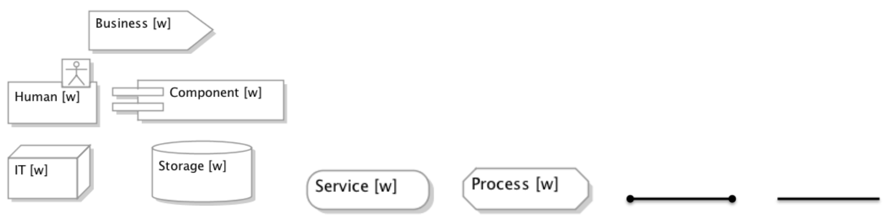
	I	II	III	IV
TOGAF [14]	No	4	No	No
ArchiMate [1]	Yes	4	No	No
URBA [10]	No	4	No	Depends on the level
SEAM [5]	Yes	User defined	Yes	Yes

Table 2. Summary of some EA approaches evaluation criteria

3 Systemic Enterprise Architecture Methodology – SEAM

SEAM is a family of methods used for consulting and teaching strategic thinking, business/IT alignment, and requirements engineering [7, 16]. In this paper we elaborate on applying SEAM in the creation of an IS cartography, so we use the Enterprise Architecture specific SEAM.

SEAM represents an organization as a **hierarchy of service systems** (from business down to IT, also known as organizational level hierarchy). To clarify, we use **system** to refer to any kind of entity in our surrounding: an organization, an employee, an IT system, or an application [17]. In General Systems Thinking (GST), the common definition of a system is "a set of elements standing in interrelations"[18] and every system (and service system) has a boundary.



(a) Systems (b) Actions (c) Refinement (decomposition) and invoke (use) link respectively

Fig. 1. SEAM notation for the different kinds of service systems, actions and links

In the mentioned hierarchy, we can model systems as a **whole, also denoted as [w] (black boxes)** or as a **composite, denoted as [c] (white boxes)**. By modeling a system as a whole, we ignore the system's components and we focus only on the services offered by the system. When we model a system as a composite, the components and their relationships are visible, so we see the implementation of the service and understand the responsibility of each component. In this case, the system seen as a composite is the parent in the organizational level hierarchy and the component systems are placed within the boundary of their parent. Essentially, there are no rules on the number of levels in the hierarchy.

Besides the organizational level hierarchy, we can specify the behavior (functionality) of each system. There are two kinds of functionalities [17]:

1. **Service (localized action)** is the behavior of a system as a whole. It represents *a service offered by a system.*
2. **Process (action binding)** is the behavior of a system as a composite and it represents *a service implementation by a system* [2]. Usually the process is connected to services from other systems as a whole (that share the same system as a composite parent).

When we have both of the views (as a whole and as a composite) for the same system in one diagram, we can observe two types of connections (links), depicted in Fig. 1(c):

– The two views are connected with a *refinement* (decomposition) link. Essentially, this link shows that it is the same system. The process in the system as a whole corresponds to the implementation of the service in the system as a composite.
– In the system as a composite, the process is connected with plain links to other services that belong to the component systems as a whole. These links mean that the process invokes (uses) those services.

As a notation, a block arrow pictogram is used to specify a general (business) service system in a SEAM diagram. There exist other pictograms that specify the nature (type) of the service system (see Fig. 1(a)). The name of the system is written on the top of the pictogram, followed by a small letter in square brackets [w] or [c], denoting if it is a system as a whole or a composite respectively. Also, the pictograms used for a service and a process are depicted in Fig. 1(b).

An example of a SEAM model is depicted in Fig. 2.

In a service oriented environment, there is no definite number of layers between the application layer and the layer where the final end user participates in/consumes the service. In Fig. 2 we show four organizational levels in a SEAM model. In this example model there is an application in the third (*Application 22*), and in the fourth level (*Application 31*). We are not showing details of the application and infrastructure layers. They can be seen after we model its composite views, but for simplicity we don't show them here.

If we try to map this model to a meta-model with fixed number of levels, we will fail. The application layer is usually fixed in one level, and we model a situation where we need an application layer in two different levels. Adding an extra application layer can solve the problem, but it also adds complexity and decreases scalability of the overall solution.

Another solution to the problem described is defining the levels to be recursive, which is possible with SEAM. If the information about the level is needed, it is possible to store it as a flag.

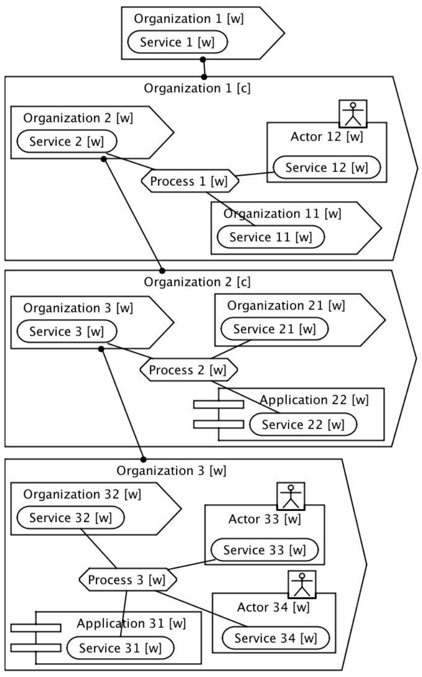


Fig. 2. Example SEAM model showing the organizational level hierarchy

4 SEAM Meta-model Implementation in Solu-QIQ

SEAM is capable of modeling more scenarios compared to other approaches, which is a great advantage. But this advantage makes it difficult to find a meta-model that every SEAM model will follow. Finding this meta-model enables modelers to create SEAM models that capture more situations in a fast way by using the Solu-QIQ tool.

The already developed meta-models, which capture all methods which are part of SEAM [11, 9] are very exhaustive. From the SEAM basic EA method explanation in Subsection 3, the meta-model should capture systems (whole and composite), actions (service and process) and connections (decomposition and usage). We also derive several modeling rules, that help in making the meta-model more simple. We summarize the rules as follows:

– A system as a whole hosts only services, and a system as a composite hosts processes and other systems as a whole.
– A process is linked with (uses) services from systems as a whole.
– The decomposition link, connects different views of a system.

From this we conclude that the only items that should be present in our meta-model are: **a system, a service** and **a process**. Our proposed meta-model is shown in Fig. 3.

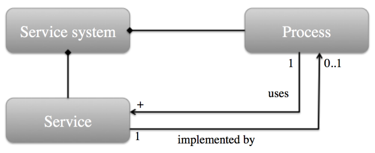


Fig. 3. Meta-model of SEAM concepts implemented in Solu-QIQ

In this class diagram of the meta-model we see two *Composition* links (**Service System** – **Process** and **Service System** – **Service**). By these two links we have partly captured the system *decomposition link*, and the *whole* and *composite* concepts. Let's say we have a service system **sysX**, that offers the service **serX** (when **sysX** is seen as a whole), and a process **procX** that implements the **serX**, which is in **sysX**, when the system is seen as a composite. Whenever a system is connected with a service, we know that we are looking at its view as a whole. The same holds for a system connected with a process and a system's view as a composite. This decomposition scenario will be complete only when we know that **procX implements serX**. This is done by the connection between Service and Process in the meta-model.

Finally, we have a many-to-many relationship between a Process and a Service. The reason for this is the invoke (use) link. The Service → Process (decomposition) link stands for "the service implemented by a process", and the Process → Service (invoke) link stands for "the process uses a service". The cardinality of processes and services is different. On the process side it is 0..* because there can be services for which we don't know the implementation. For the services it is 1..* because there can not be a process that doesn't use a service. Also, once we have a process, we have the corresponding service. The distinction between these two types of links is stored as a flag in this many-to-many relationship.

The Service System in the meta-model has one additional attribute - Type, of the SystemType. This SystemType is a typology that is used to distinguish what kind of system we want to show (see Fig. 1(a)). With this we implement the solution of having recursive layers. By giving the value *Application* to the system's type, we will know that that system is an application and belongs to the corresponding Application layer in URBA.

We tested the proposed meta-model by implementing it in Solu-QIQ. It is not intuitive that this meta-model is capable of showing infinite number of organizational hierarchy levels. For this purpose, we populated the meta-model in the tool with data that correspond to the SEAM model seen in Fig. 2. It is not enough just to insert data. It is up to us how do we interpret the meta-model and the data present. This is done by creating a query in Solu-QIQ that outputs enough data to show one organizational hierarchy level as a SEAM model. The final Solu-QIQ output is depicted in Fig. 4 and it is obtained after defining a style in the tool for the output of the query. We have to note that the output shows only one level at a time, and in this figure four separate outputs are shown.

When we compare Fig. 2 and Fig. 4, we can see that it is the same model. With this we proved that Solu-QIQ is able to generate SEAM models based on

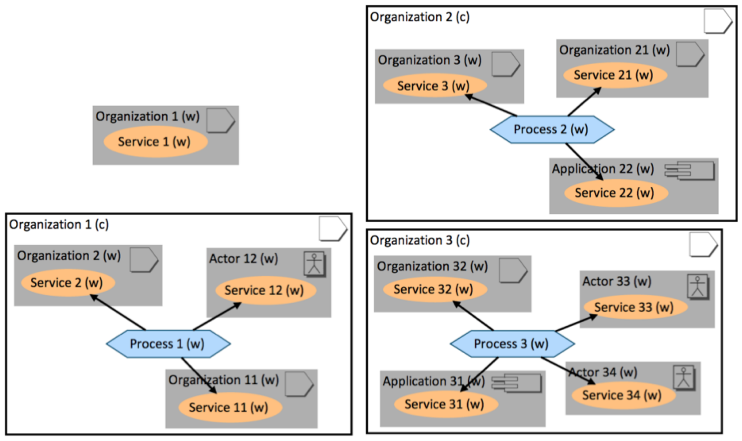


Fig. 4. A Solu-QIQ output of the SEAM model shown in Fig. 2 implemented based on the meta-model shown in Fig. 3

the meta-model we provided. So in future, instead of drawing SEAM models, system by system, action by action, we can only populate the Solu-QIQ tool with the needed data, click on a button, and we will have the model ready.

5 Conclusions

In this paper we show how to combine an EA approach with a tool independent of the approach through creating a mutually compliant meta-model. This tool is further on used for generating EA models. The overall goal is to have models of the EA that enables the IT to give better support to the enterprise's IT strategy. This is tightly connected with the definition and implementation of an IT strategy, as seen in [13].

In this paper we describe the development of a meta-model that is compliant with our chosen EA approach, SEAM, and an EA tool, called Solu-QIQ. This tool is inspired by the urbanization approach.

First, we explained why SEAM and a cartography tool such as Solu-QIQ are needed in an enterprise, and why it is important for them to coexist. Then, we gave a simple meta-model that captures the basic SEAM principles. We finish with the implementation of this meta-model in Solu-QIQ and show the output of a generated model. By doing all this, we showed the recursive side of SEAM and we display the efficiency of Solu-QIQ when dealing with recursive meta-models.

6 Future Work

Our next step is adding a concept in the meta-model that shows grouping of services by functionality. As SEAM is a service-oriented approach, this grouping will also give the service catalog.

The presented meta-model captures only the basic SEAM usage and capability. SEAM is much more powerful than showing services, processes, service systems as a whole and as a composite. In order to benefit from the complete SEAM approach, this meta-model has to be augmented with the other SEAM concepts not presented here.

References

1. ArchiMate, an Open Group Standard. http://www.opengroup.org/subjectareas/enterprise/archimate.
2. B. Bajić-Bizumić, C. Petitpierre, H. C. Huynh, and A. Wegmann. A model-driven environment for service design, simulation and prototyping. In *Exploring Services Science*, pages 200–214. Springer, 2013.
3. BeLINK. http://www.groupe-belink.fr/.
4. Enterprise Architect. http://www.sparxsystems.com/products/ea/.
5. A. W. et al. Requirements Modeling in SEAM: The Example of a Car Crash Management System. In *Comparing Requirements Modeling Approaches (CMA@RE) workshop*, 2013.
6. Iteraplan. http://www.iteraplan.de/en/eam-iteraplan.
7. LAMS. Seam for Teaching. http://lams.epfl.ch/reference/seam/.
8. M. Lankhorst. *Enterprise Architecture at Work: Modelling, Communication and Analysis.* Springer, 2009.
9. L.-S. Lê and A. Wegmann. Hierarchy-oriented modeling of enterprise architecture using reference-model of open distributed processing. *Computer Standards & Interfaces*, 35(3):277–293, 2013.
10. C. Longépé. *The Enterprise Architecture IT Project: the Urbanisation Paradigm.* Butterworth-Heinemann, 2003.
11. I. Rychkova. Formal semantics for refinement verification of entreprise models. *Unpublished doctoral dissertation, University of Lausanne, Switzerland*, 2008.
12. R. Sessions. Comparison of the Top Four Enterprise Architecture Methodologies, 2007.
13. G. Tapandjieva, D. R. Marchetti, I. Rychkova, and A. Wegmann. Towards the Definition, Implementation and Communication of an IT Strategy: the Case of IT Strategy at EPFL. In *The 8th International Workshop on Business/IT-Alignment and Interoperability*, 2013.
14. The Open Group Architecture Framework (TOGAF). http://www.opengroup.org/subjectareas/enterprise/togaf/.
15. L. Urbaczewski and S. Mrdalj. A Comparison of Enterprise Architecture Frameworks. *Issues in Information Systems*, 7(2):18–23, 2006.
16. A. Wegmann. On the Systemic Enterprise Architecture Methodology (SEAM). In *Published at the International Conference on Enterprise Information Systems 2003 (ICEIS 2003).* Citeseer, 2003.
17. A. Wegmann, A. Kotsalainen, L. Matthey, G. Regev, and A. Giannattasio. Augmenting the Zachman Enterprise Architecture Framework With a Systemic Conceptualization. In *Enterprise Distributed Object Computing Conference, 2008. EDOC'08. 12th International IEEE*, pages 3–13. IEEE, 2008.
18. G. M. Weinberg and J. Wiley. *An Introduction to General Systems Thinking.* Wiley New York, 1975.