2007

An example of a hierarchical system model using SEAM and its formalization in Alloy

Alain Wegmann
Ecole Polytechnique Fédérale de Lausanne

Lam-Son Le
Ecole Polytechnique Fédérale de Lausanne, lle@uow.edu.au

Jose Diego de la Cruz
Ecole Polytechnique Fédérale de Lausanne

Irina Rychkova
Ecole Polytechnique Fédérale de Lausanne

Gil Regev
Ecole Polytechnique Fédérale de Lausanne

Publication Details
An example of a hierarchical system model using SEAM and its formalization in Alloy

Abstract
SEAM is an enterprise architecture method based on RM-ODP part 2. In this paper, we present an example of a SEAM hierarchical model where the behavioral part is formalized in Alloy. We introduce four kinds of actions and their relationships to properties that specify object state. We show that the Alloy formalization enables us to check that the SEAM model conforms to the required aspects of the universe of discourse.

Disciplines
Physical Sciences and Mathematics

Publication Details

This conference paper is available at Research Online: http://ro.uow.edu.au/infopapers/3686
An Example of a Hierarchical System Model Using SEAM and its Formalization in Alloy

Alain Wegmann, Lam-Son Lê, José Diego de la Cruz, Irina Rychkova, Gil Regev
Ecole Polytechnique Fédérale de Lausanne (EPFL)
School of Computer and Communication Sciences
CH-1015 Lausanne, Switzerland
{alain.wegmann, lamson.le, josediego.delacruz, irina.rychkova, gil.regev}@epfl.ch

Abstract

SEAM is an enterprise architecture method based on RM-ODP part 2. In this paper, we present an example of a SEAM hierarchical model where the behavioral part is formalized in Alloy. We introduce four kinds of actions and their relationships to properties that specify object state. We show that the Alloy formalization enables us to check that the SEAM model conforms to the required aspects of the universe of discourse.

Keywords: RM-ODP, Enterprise Architecture, Ontology, Formalization, Alloy.

1. Introduction

SEAM is an Enterprise Architecture method with which we systematically and systemically analyze and design business and IT systems [1]. SEAM stands for “Systemic Enterprise Architecture Method”. Within SEAM we consider a company of interest, its organization and its environment as a hierarchy of systems. We typically identify four levels of systems: the market segment, the company’s value network, the company’s organization and the company IT system. We then analyze and design all these systems. SEAM has been used for teaching [2] and consulting [3] for the last few years.

The SEAM modeling ontology is based on RM-ODP Part 2 [4]. We purposely do not use the viewpoints introduced in RM-ODP part 3 as these viewpoints were defined to specify different views of an IT system. They are therefore IT system centric. In SEAM, we consider the IT system as one of many other systems (e.g. the company, the company value network or the market segment). Inspired by Systems Thinking [5], we use the same kind of representation for the different kinds of systems. Systems Thinking is a discipline in which the analogies between systems are studied regardless of the difference in nature of these systems. The SEAM modeling ontology was initially presented in [6] and is more precisely described in [6]. More specifically, in [6] we introduce four kinds of actions that are necessary to represent system behavior. This paper extends [6] by providing an Alloy example that illustrates the specificities of these four kinds of actions. The paper also introduces an experimental graphical notation that can represent the pre, post conditions and invariants necessary to define the actions.

Alloy [7] is a lightweight formal specification language with which we build logical models that can ultimately be verified for consistency by an accompanied tool called Alloy Analyzer. An Alloy model is composed of: signatures that define sets and relations; facts and predicates that define global constraints, invariant properties; functions that define parameterized constraints; assertions that define properties to check; command that run functions and/or check assertions. Once it is expressed in Alloy, a model can be verified by having the Alloy Analyzer generate snapshots with sample instances. If the Analyzer cannot create a snapshot, the specification might contain contradictions. This is a first kind of verification that can be done. If a snapshot is created, it can be visually analyzed by the modeler to check whether the model has the expected properties. For example, if a modeler specifies an acyclic graph, she can check the snapshots to ensure that no cycles are present.

In Section 2, we introduce some of the concepts of the SEAM ontology. In Section 3 we present an example of an on-line bookstore based on the concepts defined in Section 2. The example is represented both graphically and with verified Alloy code. In Section 4, we highlight some of the features of the approach. In Section 5, we present relevant related work. In Section 6, we conclude and propose future research directions.
2. SEAM Modeling Ontology

In this section, we describe the minimum concepts necessary to model system behavior. We introduce the terms working object, environment, property and action. We define four types of actions and explain the relationships between actions and properties in terms of pre, post-conditions and invariants.

A [working] object\(^1\) is defined in RM-ODP part 2 [4] as the model element that represents a system in the universe of discourse. RM-ODP part 2 [4] defines a system as an entity that can be considered as a whole or as comprised of parts. A working object, therefore, can be represented either as whole or as composite (i.e. comprised of parts). Both representations coexist in a model. A working object, represented as a whole, exhibits properties and partial actions that modify the state of these properties. A working object, represented as a composite, exhibits component working objects (exhibiting properties) and full actions that modify the state of these properties. A working object can be drawn with different pictograms depending on the kind of system it represents. In this paper, we draw working objects as block arrows because they represent business entities such as a value network or a company. Block arrows are frequently used to represent business entities [8]. For example, in Fig. 1 (a), the working object S is represented as a whole. It is named S_W, with “W” meaning “whole”. In Fig. 1 (b), it is represented as a composite; it is named S_C (with “C” for composite).

Quite frequently it is useful to make an abstraction of the working objects that interact with a working object of interest. All the abstracted objects are represented by what we call the environment of the object of interest. This definition is compatible with RM-ODP part 2 that states that the environment is “the part of the model which is not part of [the] object [of interest]” [4]. In the SEAM graphical representation, the environment is drawn as a grayed area that surrounds the object of interest (see Fig. 1).

System behavior, in SEAM, is described by properties and actions. RM-ODP part 2 defines the concept of object state. We add the concept of property, which contain the state of the object. This is necessary to structure the state and to be able to specify the effects of the object’s actions. Properties are modified by actions. Properties are drawn as rectangles. The name of a property is prefixed by cardinality information. For example, in Fig. 1 (a), exactly one property SP1 exists in the context of S_W (i.e. system S considered as a whole).

As defined in RM-ODP part 2, an action is a model element that represents something that happens in a system, i.e. a change to one or more properties of a working object. Actions are drawn as rounded rectangles. We define two characteristics for actions: full/partial and local/non-local. The full/partial characteristic depends on the kind of representation of the working object in which it is defined. A partial action\(^2\) is defined in a working object as a whole. In Fig. 1 (a), T and U are partial actions or localized actions. A full action\(^3\) is defined in a working object as a composite. It involves all or part of the component working objects. In Figure 1(b), V and W are full actions. The local/non-local characteristic specifies whether the working object’s environment participates in the action. A local action does not involve the environment. For example, the actions U in Fig. 1 (a) and W in Fig. 1 (b) are local. A non-local action does involve the environment e.g. actions T and V. In contrast, RM-ODP part 2 defines only the notions of local action and interactions. Fig. 2 illustrates the relation between the characteristics we have defined and the RM-ODP part 2 terms.

\(^1\) We added the term “working” to avoid confusions with the other use of the term object (e.g. object-oriented programming).

\(^2\) Note that the concepts of localized actions and joint actions were made popular by Catalysis [9]. We consider them as synonym to partial actions and full actions.

---

Fig. 1: Example of the notation.
In order to model the behavior of a system, we need to precisely define the relationships between the actions and the properties of the working objects. We therefore define pre-conditions, post-conditions and invariants for each action. They are graphically represented on the associations that connect the actions to the properties. An outgoing navigability from an action into a property indicates that the property is a post-condition of the action. An incoming navigability from a property into an action indicates that the property is a pre-condition of the action; no navigability indicates an invariant. Each action/property association is an invariant, a pre or a post-condition. The pre-conditions, post-conditions and invariants are written using the Alloy language, as illustrated in Section 3.

SEAM is developed for the alignment of business and IT. The originality of SEAM is its capability to use the same method for the design of a marketing strategy, an outsourcing strategy, an organizational strategy, or the IT system itself. We therefore analyze and design a hierarchy of systems, starting from the environment of the company of interest, i.e. the market segments in which it operates. A market segment, a value network, a company organization, and IT applications are all considered as systems and modeled with working objects. In the example presented in this paper, we model a simplified market segment (in which we omit the competition) as well as the value network to which the company of interest belongs. A value network is a group of companies that provide a service to a customer. It resides in a market segment.

3. Example of Hierarchical Modeling

We present the example of an enterprise model that describes an on-line bookstore. The model contains four views. Two views specify the service provided by the company value network to its customers. Two other views analyze how this service is provided by the companies involved in the value network.

Additional views can be added to represent the internal configuration of the company of interest and the company’s IT systems. This is not illustrated in this paper. More information can be found in [3].

3.1. Specification of the SaleAction in the Segment as a Composite

The first view (Fig 3) represents the market segment. We specify saleAction – i.e. the exchanges between the bookstore value network and the customer. The bookstore value network is considered as a whole. This hides the details such as the responsibilities of the companies that belong to the value network. This is useful to model the overall customer experience.

saleAction is a local full action. It is a full action because it is executed in a working object as a composite: BookstoreSaleCustomerSegment_C. As such it involves more than one participant working object BookstoreValueNetwork_W and Customer_W. The saleAction accesses and modifies properties that exist in BookstoreValueNetwork_W and in Customer_W. It is local because the environment of BookstoreSaleCustomerSegment_C is not affected by the occurrence of the action. The market segment in this example is considered as a closed system.

From the model in Fig. 3 we generate the Alloy code in Fig. 4. In Alloy, sets of elements are defined using the keyword “sig” (for signature). Fields can be defined in each signature, A field represents a relation between set elements. For example, the working object BookstoreValueNetwork_W is represented as a signature which contains four fields. Each field corresponds to a property defined in the working object. We include cardinality information in field definitions. In our example the BookstoreValueNetwork_W refers to exactly one BookstoreSaleCustomerSegment_C but its inventory refers to a set of Book. Note that the keyword “set” is written before the relation symbol (i.e. the arrow) to specify the cardinality of the books relatively to the relation (i.e. many books can be associated to one Time element).
Fig. 3: SEAM declarative specification of the Sale local full action in BookstoreSaleCustomerSegment_C.

sig BookSpec
{ pn: one PartNumber, price: one Int }  
sig Book
{ spec: one BookSpec }  
lone sig BookstoreSaleCustomerSegment_C {  
  bookstore: one BookstoreValueNetwork_W,  
  customer: one Customer_W  
}  
lone sig BookstoreValueNetwork_W {  
  segment: one BookstoreSaleCustomerSegment_C,  
  catalog: set BookSpec,  
  inventory: Book set -> Time,  
  cash: Int one -> Time  
}  
lone sig Customer_W {  
  segment: one BookstoreSaleCustomerSegment_C,  
  wantedPN: one PartNumber,  
  bookshelf: Book set -> Time,  
  cash: Int one -> Time  
}  

pred saleAction[  
  aSeller: one BookstoreValueNetwork_W,  
  aBuyer: one Customer_W,  
  pre: one Time, post: one Time] {  
  // inv1  
  one bs: BookSpec |  
  (bs.pn = aBuyer.wantedPN) and  
  (bs in aSeller.catalog)  
  // pre1  
  one bk : Book |  
  (bk.spec.pn = aBuyer.wantedPN) and  
  (bk in aSeller.inventory.pre) and  
  (bk not in aBuyer.bookshelf.pre)  
  // pre2  
  one bs: BookSpec |  
  (bs.pn = aBuyer.wantedPN) and  
  (int aBuyer.cash.pre >= int bs.price)  
  // post1  
  one bk : Book |  
  (bk.spec.pn = aBuyer.wantedPN) and  
  (aSeller.inventory.post = (aSeller.inventory.pre - bk))  
  and  
  (aBuyer.bookshelf.post = (aBuyer.bookshelf.pre + bk))
Fig. 5 defines the invariants, pre-conditions and post-conditions of `saleAction`.

The invariant of `saleAction` is:
- `inv1`: one `BookSpec bs` exists such that (a) the book spec has the part number specified by the buyer and (b) the seller catalog contains the book spec.

To execute `saleAction`, the following pre-conditions must be satisfied:
- `pre 1`: one `Book bk` exists such that (a) the book has the part number specified by the buyer and (b) the book is in the seller inventory and (c) the book is not already in the buyer bookshelf.
- `pre 2`: one `BookSpec bs` exists such that (a) the book spec has the part number specified by the buyer and (b) the buyer has more cash than the price specified in the book spec.

The `saleAction` post-conditions are:
- `post1`: one `Book bk` exists such that (a) the book has the part number specified by the buyer and (b) the book does not exist anymore in the seller inventory and (c) the book exists in the buyer bookshelf.
- `post2`: one `BookSpec bs` exists such that (a) the book spec has the part number specified by the buyer and (b) the price specified in the book spec is added to the seller’s cash.
- `post3`: one `BookSpec bs` exists such that (a) the book spec has the part number specified by the buyer and (b) the price specified in the book spec is deducted from the buyer’s cash.

When the code defined in Fig. 4 and Fig. 5 (plus a few additional lines necessary for technical reasons) is fed into the Alloy analyzer, the tool generates snapshots that show possible states of the system before and after the execution of the `saleAction`. Fig. 6 and Fig. 7 illustrate the kinds of snapshots generated. When comparing Fig. 6 and 7, it is possible to see that `Book1` moves from the inventory of `BookstoreValueNetwork_W` to the bookshelf of `Customer_W`. On the other hand, `7` moves from the `Customer_W cash` to the `BookstoreValueNetwork_W cash`. This corresponds to the expected behavior of our model.

3.2. Specification of the `sellAction` of the Value Network as a Whole

The second view (Fig. 8) focuses on the bookstore value network and abstracts away the customer. We model `sellAction` that represents the contribution of the bookstore value network in the `saleAction` previously defined.

The `sellAction` is a non-local partial action. It is a partial action because it is executed by a working object as whole: `BookstoreValueNetwork_W`. It is a non-local action because the environment of `BookstoreValueNetwork_W` is involved in the execution of the action.
The Alloy model that describes the sellAction is in Fig. 9.

```alloy
lone sig SellEnvironment {
    spec: one BookSpec,
    out_book: Book lone -> Time,
    in_cash: Int one -> Time
} {…)
}

pred sellAction[
    aSeller: one BookstoreValueNetwork_W,
    aEnv: one SellEnvironment,
    pre: one Time, post: one Time
] {
    // inv1
    one bs: BookSpec | (bs.pn = aEnv.spec.pn) and
    (bs in aSeller.catalog)

    // pre1
    some bk: Book | (bk.spec = aEnv.spec) and
    (bk in aSeller.inventory.pre) and
    (no aEnv.out_book.pre)

    // pre2
    one bs: BookSpec | (bs = aEnv.spec) and
    (int aEnv.in_cash.pre >= int bs.price)

    // post1
    some bk: Book | (bk.spec = aEnv.spec) and
    (aSeller.inventory.post =
    (aSeller.inventory.pre - bk))
    and
    (aEnv.out_book.post = bk)

    // post2
    one bs: BookSpec | (bs = aEnv.spec) and
    (int aSeller.cash.post =
    (int aSeller.cash.pre +
    int bs.price))

    // post3
    one bs: BookSpec | (bs = aEnv.spec) and
}
```

Fig. 9: Alloy declarative specification of sellAction non-local partial action in BookStoreValueNetwork_W

It is interesting to compare the sellAction (Fig. 9) with the saleAction (Fig. 5). The Alloy model structure is very similar, with the difference that the Customer_W is replaced by SellEnvironment. SellEnvironment represents what is relevant from the environment of BookstoreValueNetwork_W when a sellAction is executed. The specification of the environment depends on the working object of interest (which is obvious) but also on what the object executes (i.e. the sellAction). The specification of the environment reflects precisely the specification of the system of interest.

3.3. Specification of the MarketAndShipAction on the Value Network as a Composite

The third view (Fig. 10) describes the bookstore value network as a composite. It describes how the companies PublisherCompany and ShippingCompany interact to serve the customer. We model the marketAndShipAction. This action can be considered as the implementation of the sellAction.

The marketAndShipAction is a non-local, full action. It is a full action because it is executed by a composite working object BookstoreValueNetwork_C. The PublisherCompany_W and ShippingCompany_W participate in the action. It is a non-local action because there are exchanges with the environment of BookstoreValueNetwork_C (for example interacting with the customer). So BookstoreValueNetwork_C is not modeled as a closed system.

As discussed in 3.1, we need to specify in Alloy the structure of the working object as a composite. This is done in Fig. 10 and the equivalent Alloy model is in Fig. 11. The BookstoreValueNetwork_C is composed of a PublishingCompany_W and ShippingCompany_W. If we compare Fig. 11 and Fig. 4, we notice that there are similarities. The same fields are defined but they are now distributed in the component working objects.
For example, the inventory defined in the BookstoreValueNetwork_W (Fig. 4) is now defined in PublisherCompany_W as a whole (Fig. 11). Additional fields are added to capture the interaction between the companies. For example, ShippingInfo represents the information exchanged between PublisherCompany_W and ShippingCompany_W.

Once the structure of the Alloy model is defined, we need to specify the marketAndShipAction (Fig. 12) If we compare the specification of marketAndShipAction with the sellAction (Fig. 9), we notice similarities. The Alloy model has a similar structure; the difference is the location of the fields and the addition of the exchange of information between the companies (such as the shipping info). As marketAndShipAction is a non-local action, it modifies MarketAndShipEnvironment that represents the environment of BookstoreValueNetwork_C.

Fig 11: Alloy declaration of the working objects BookstoreValueNetwork_C, PublisherCompany_W, and ShippingCompany_W.
Fig 12: Alloy declarative specification of MarketAndShip non-local full action in BookstoreValueNetwork_C

4. Discussion

With the example in Section 3, we have hinted how the alignment between business and IT can be achieved. We have defined the service offered to a customer (saleAction). We then map this service to the role for a group of companies (sellAction of BookstoreValueNetwork_W). This role is translated into an implementation that involves several companies (marketAndShipAction of BookstoreValueNetwork_C). This illustrates that even if RM-ODP was developed for specifying IT systems it can also be used to specify pure business systems.

To be able to maintain the traceability between these views, we need the four kinds of actions we have introduced in Section 2 and illustrated in Section 3. The design begins with a local full interaction (saleAction) that specifies the overall business goal. Then a non-local partial interaction (sellAction) is defined. sellAction specifies the responsibility of the company of interest together with its partners. At that point a non-local full interaction (marketAndShipAction) is defined to specify the implementation. As illustrated in the example, the specifications of these different actions have strong similarities and it is conceivable to provide tool support for the design process.

In our example we have concretely illustrated the differences between these actions. The difference between a local and a non-local action resides in the presence of an environment in the specification of the action. The difference between a full and a partial action resides in the way the Alloy model is written. The model for the full action refers to multiple working objects. The model for a partial action refers to one working object only.

We have also illustrated the notion of environment that depends of what is the system of interest and what is its behavior.

5. Related Work

Our work does not make references to the RM-ODP viewpoints. We directly model systems using the terms defined in RM-ODP part 2. This point can be argued and it can be considered that, depending on the universe of discourse, we actually describe viewpoints. For example, in Section 3, we can consider that we describe an enterprise viewpoint. In addition, as we define pre and post conditions, we define an information viewpoint as well. The RM-ODP viewpoints were designed to represent different views necessary to specify an IT system and they are not really hierarchical. As our main focus is the alignment between business and the IT, we choose to model a hierarchy of systems using an ontology designed to maintain the traceability between views.

Our approach is based on the transformation of joint actions into localized actions and vice-versa. This idea was made popular by Catalysis [9]. Our method can be considered as a possible RM-ODP-based implementation of Catalysis. It is also important to highlight that the concept of joint action was first introduced in [10] and then further developed in Disco3.

In [11] Sinderen and al. propose an RM-ODP design method that has perspectives that are similar to the views presented in Section 3.

Other hierarchical system modeling methods include the following: Kobra [12] is close to the Unified Modeling Language (UML). If a similar approach would be defined for Kobra, it would use the UML meta-model and OCL instead of RM-ODP and Alloy. However, Kobra models are not verified incrementally, as in SEAM. SysML is the OMG initiative to model systems. SysML is not strictly hierarchical and the nature of the components change for each level of description. SysML is based on UML. A more detailed comparison between UML and SEAM is available in [13]. Other methods similar to SEAM are OPM [14], Adora [15] and Demo [16]. These methods are not based on RM-ODP.

5 SysML http://www.sysml.org/
6. Conclusion & Future Work

In this paper, we presented an example of hierarchical modeling with SEAM, directly based on the concepts defined in RM-ODP part 2. We introduced four kinds of actions, in two dimensions (partial/full and local/non-local). We also propose a graphical notation and a formalization with the Alloy specification language. Our goal for this paper was not to present how SEAM can be applied in concrete projects. Interested readers can find more information on SEAM’s applicability in [8].

The notation and the Alloy formalization presented in this paper contain many details that are not yet used for the development of concrete enterprise models. Nevertheless, these are necessary for the development of simpler notations that can be used in concrete projects.

Future work includes: (a) the definition of declarative semantics for SEAM and of the interpretation we made of RM-ODP part 2 beyond the example provided in this paper. (b) the development of an Alloy model that represents imperative concepts such as activities, including constraints between actions. (c) the exploration of tool support for design process. The example has shown that the specifications of the four kinds of actions are very close to each other. It would be interesting to provide mechanisms to support the development and the validation of these specifications.

7. References