Operational ASM Semantics behind Graphical SEAM Notation

Irina Rychkova, Alain Wegmann[1], and Pavel Balabko[1]

School of Computer and Communication Sciences (I&C), École Polytechnique Fédérale de Lausanne (EPFL) CH-1015 Lausanne, Switzerland
{Irina.Rychkova, Alain.Wegmann, Pavel.Balabko}@epfl.ch
http://lamswww.epfl.ch

Abstract. The context of this paper is Enterprise Architecture (EA). EA is a multi-disciplinary approach that allows different specialists to design new business and IT systems and focuses on the integration of these systems. Our group develops a specific EA method that is called SEAM. The current version of SEAM has a formal denotational semantics for its modeling language. In order to provide model simulation and checking at each level of abstraction, SEAM needs an operational semantics. Currently this work is at the stage of problem setting. In this paper[1]we present SEAM and describe the main research problem. We propose to use ASM as operational semantics for SEAM to verify that models, produced by different specialists are consistent. We illustrate our approach by giving an example of SEAM notation that has already been mapped to ASM.

1 Introduction

Enterprise architecture (EA) is a multi-disciplinary approach that enables enterprises to anticipate or react to necessary business or technical changes. The EA team designs and deploys new organizations and IT systems in the light of necessary changes. In an EA project, the EA team develops a model that represents the enterprise: the enterprise model. The enterprise models are usually structured in hierarchical levels. The highest level describes the marketing aspects, the middle level describes the business processes, and the lower level describes the IT systems.[1]

Our group develops a theory for enterprise architecture (EA) and then applies this theory to the development of a specific EA method called SEAM [1]. SEAM stands for the "Systemic Enterprise Architecture Methodology" or for seamless integration between business and IT.

The important parts of SEAM are *the method* and *the notation*. The SEAM method explains how to proceed in the analysis and design of the enterprise[1].

[1] This paper is written by Irina Rychkova. Alain Wegmann, the scientific advisor of the project, proposed to study the EA context. This work is a continuation of the work done by Pavel Balabko, who proposed to consider ASM in context of our research problem.

The SEAM notation defines a graphical modeling language. This work mainly deals with the SEAM notation. In context of EA, graphical notation is essential. Graphical model representations can be less ambiguous and much more efficient for communication than plain text. As EA projects involve specialists from different disciplines, it is important to provide means to simulate the model (to ease communication between the specialists) and means to check the model (to verify that the different levels developed by the different specialists are compatible). For this purpose, in addition to graphical notation which is defined by a denotational semantics, we need to provide an operational semantics for our modeling language.

In this paper we present our work that is currently at the stage of problem setting. Our project has 3 main goals: to define more precisely the SEAM modeling language, to provide an operational semantics for this language and to validate the impact of having an operational semantics in context of EA. For this paper we focused mostly on operational semantics for SEAM in order to provide model simulation and checking.

In section 2 of this paper we consider the SEAM method and its main aspects in the context of EA. In section 3 we formulate our research problem. In the first part of section 4 we justify the choice of ASM as a solution of our research problem and observe the advantages of this semantics. Then, in the second part of section 4, we consider how to define the SEAM notation in ASM: we discuss a tool we plan to develop and illustrate the modeling process on the example. Section 5 is the conclusion.

2 SEAM Method in Context of Enterprise Architecture

The goal of SEAM as a modeling language is to serve as a uniform notation for all enterprise stakeholders that participate in the modeling process. While developing the SEAM notation we are trying to be as close as possible to UML. At the same time, SEAM has several characteristics which, we believe, make it more appropriate than UML for EA modeling.

SEAM: A Method for Stepwise Design Using Hierarchical Models

SEAM enterprise model describes a hierarchy of systems, which includes the business and IT resources together with the processes in which they participate. **Hierarchical Model:** The enterprise model is typically structured in levels (business, operational, IT)[2]. In SEAM, at each level, systems of interest can be represented with a computational viewpoint (CV), as a collaboration of subsystems. At the same time, each subsystem can be described with an information viewpoint (IV) (Fig. 1). This is inspired by RM-ODP.[7]

The computational viewpoint (CV): a viewpoint on the system of interest that enables distribution through the functional decomposition of the system into subsystems that interact at interfaces. CV is concerned with the description of the system as a set of physical objects.

The information viewpoint (IV): a viewpoint on the system of interest that focuses on the semantics of the information and information processing performed. IV is concerned with the information that needs to be stored and processed in the system and describes the behavioral aspects of the system.

As a result of the modeling process, we can build the hierarchy of CV specifications from whole to composite (collaboration of subsystems), (Fig. 1) and hierarchy of IV specifications from general to detailed (Fig. 2).

Stepwise design: SEAM method realizes recursive modeling of a hierarchical system. There is a universal modeling template that could be described as:

1. Take a CV of the system at the highest level (Fig. 1-(a) and Fig. 2-(a) that represent the same system "System1").
2. Make an IV specification for the sub-system(s) of interest. The IV specification(s) includes system policies and non-functional requirements documented as assumptions (Fig. 2-(b) that represents "subS1" sub-system of "System1" together with a set of assumptions).
3. Make a detailed IV specification for the sub-system(s) of interest. The detailed IV specifications transform the assumptions into behavior (Fig. 2-(c)that represents "subS1" information viewpoint with all necessary details for its implementation).
4. Define the CV at the next level by making a CV refinement of the system of interest (Fig. 1-(b) that represents "subS1" computational viewpoint corresponding to the information viewpoint of "subS1" in Fig. 1-(a)).
5. Make an IV specification for the subsystem(s) of interest (step 2).
6. Make a detailed IV specification for the sub-system(s) of interest (step 3).
7. Iterate steps 4 to 6 for all CV levels.
8. Verify that the model is complete and coherent.

It is interesting to highlight that SEAM is an evolution of Catalysis [11]. Some of the Catalysis originalities (that are kept in SEAM) are: the hierarchical system design (across 3 levels) and the concepts of joint actions to specify the system goals. SEAM keeps these features and adds a more precise ontology based on RM-ODP, the possibility to design more than 3 levels, and a notation better suited for system representation.

SEAM: A Notation for System Modeling

Modeling an enterprise across its levels is difficult. To make it practical, it is important to have a modeling notation well suited for system modeling. To illustrate this, we give one example which is the integration of the system's behavior and the system's information representation. Generally speaking, the UML diagrams can be categorized either as structural or as behavioral diagrams. However, system theory has shown that the separation behavior / structure is artificial [6] and actually prevents the development of models that truly represent the changes happening in the modeled system. For example, in UML, it is not possible to show in one diagram that an action changes the value of an attribute. That can make the diagram difficult to understand. This point has been

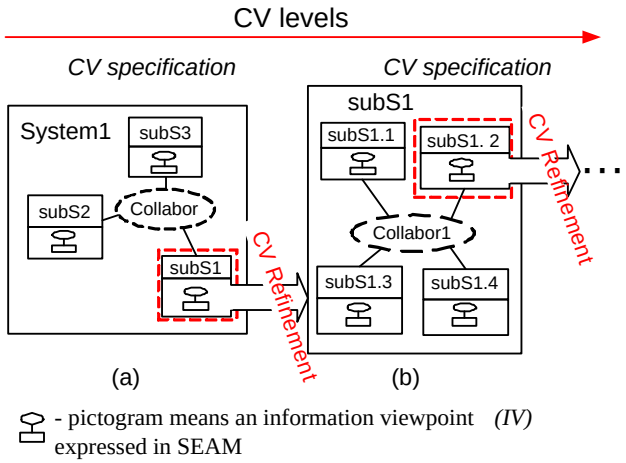

Fig. 1. SEAM hierarchy of subsystems. a) CV specifications of a System1 modeled as a collaboration of subsystems (composite view). Subsystem 1 (subS1) modeled as a whole. b)Subsystem 1 (subS1) is modeled as a composite (collaboration of subS1.1, subS1.2, e.t.c.).

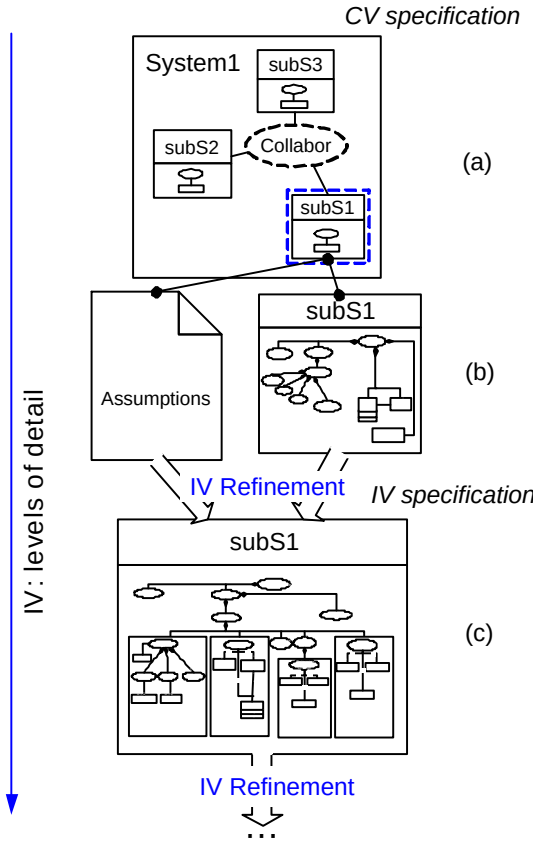

Fig. 2. SEAM levels of details. Top part of the figure shows the CV specification of the System1 that contains the very general IV specification of subsystem 1 (subS1). Middle and bottom parts of a figure show the result of subS1 model refinement. This set of IV specifications from general to detailed makes up a hierarchical model of system behavior.

identified by OPM language developers [4]. SEAM also proposes the solution for this problem.

3 Main Research Problem: Operational Semantics for SEAM Modeling Language

Denotational semantics and operational semantics, being transferred from the context of programming languages, play important roles in the definition of the modeling languages. *Denotational semantics* provide mathematical models and define relations between the terms of a modeling language. *Operational semantics* is essential for modeling languages when their applications are supposed to be simulated on a machine.

The current version of denotational semantics for the SEAM modeling language is based on [3]. In order to provide model simulation and checking, SEAM needs to have an operational semantics.

The possibility to *simulate* the model is the best way for people to figure out what is actually represented in the model. The possibility to *check* models is important for the comparison of models.

In our work we propose to map the SEAM notation into defined formal notation for which simulation and model checking tools have been developed. A good choice for these purposes is an Abstract State Machine (ASM) notation [5]. Note that the project will also have to define more precisely the SEAM modeling language and will evaluate the effect of the proposed approach by applying it to models existing in EA.

4 Abstract State Machines and SEAM Method

ASM is a method of stepwise refinable abstract operational modeling [5]. An ASM model can be used to capture the abstract structure and behavior of a discrete system.

4.1 Abstract State Machines as Operational Semantics for SEAM

In this work we propose to use ASM as an operational semantics for the SEAM modeling language. It is possible to talk about SEAM-ASM conformity for several reasons:

- In the context of requirements engineering it is important to have an abstract specification of a system without mention of its implementation. For users, in SEAM methodology a system can be represented with an information viewpoint (IV). At any level of details a SEAM IV specification can be described by an ASM specification.
- Both SEAM and ASM support principles of hierarchical system design. In ASM the hierarchy of intermediate models can be constructed by stepwise refinement (or adding more details) to the model. An ASM program can be

executed at any level of details. It corresponds to the SEAM hierarchy of IV specifications across level of details;
– In SEAM models we describe a system at any time as a pair (state, behavior), as well as in ASM. State is defined by a number of attributes and their current values. Behavior is defined as a set of actions that change a system state.

Using ASM as an operational semantics for SEAM we can obtain the following advantages for modeling:

Model Simulation. SEAM can represent models hierarchically, with several levels of detail (one of its benefits). Each IV specification in SEAM can be represented and simulated by ASM. (Fig. 3-a) This allows us to use ASM models as test models (to be matched by all stakeholders).

Refinement Checking. SEAM-ASM integration helps to make smooth and correct refinements. On each level we extend the system functionality and can also change its structure (redefine previous set of states). But each next level should correctly simulate the higher level model.(Fig. 3-b)

Model Validation and Version Comparison. ASM + verification tool = model validation and the comparison of alternative models, testing deadlocks and forbidden parameter combinations, generating of test sequences and possible sequences of states (functionality checking).(Fig. 3-c)

4.2 SEAM notation at ASM

We intend to develop an environment for our modeling techniques in order to gain practical benefits from its application. Environment can be divided into graphical, simulation, and verification tools.

Graphical tool. Graphical specifications are basic elements of our method. The Graphical tool is supposed to provide drawing and storing of SEAM models.

Simulation and verification tool. The ASM method has a tool support for simulation and verification of its models. Specifically, in [9] AsmL is presented. Another ASM tool environment for model simulation and verification is ASM Workbench [8]. In our tool we intend to use one of these tools by creating an interface or translator from SEAM to ASM notation.

Illustration of a Modeling Process (Vending Machine Example). To illustrate how our method works, we propose an example based on the Vending Machine (VM) case study [10] and its ASM specification, written in AsmL.[2]

[2] In this work we use an ASM-based specification language tool called Abstract State Machine Language or AsmL.[9] It is a language for modeling the structure and behavior of discrete dynamic systems based on the ASM method. AsmL specifications may be executed as programs. (tool support: AsmL tool version 2.0, developed by the Microsoft research group.)

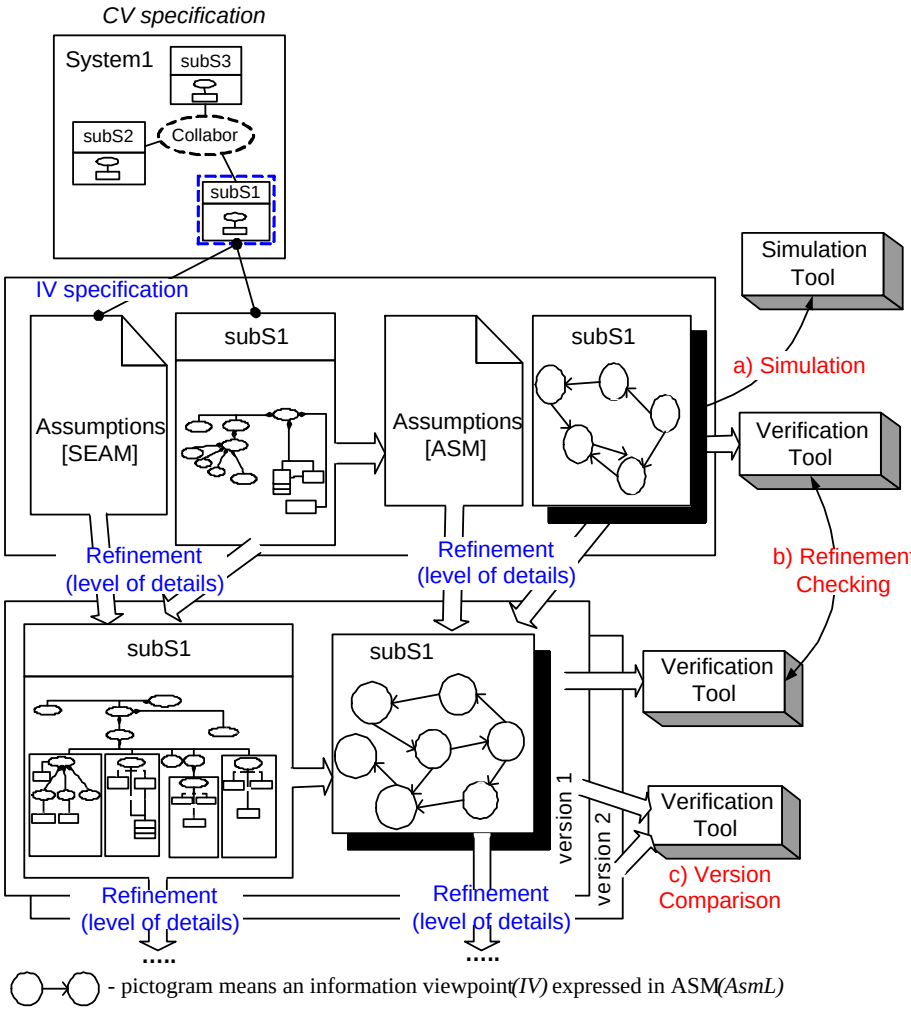

Fig. 3. ASM as an operational semantics for SEAM: a) The ASM method provides the simulation of SEAM models (IV specifications) on every level of details; b)using a verification tool for ASM models at different levels of details helps to make a correct model refinement; c) using a verification tool for different version of SEAM-ASM models allows us to make a model validation and version comparison.

VM Description. *A Vending Machine*(VM)[3] accepts money and, if there is sufficient credit, dispenses items selected by a customer. Items are identified by an alphanumeric code, called the selection number. Customers select items by entering the selection number via a keypad.

Studying the VM example we focused on the mapping between SEAM and ASM concepts. We started with an initial IV graphical specification for a model (Fig. 4), based on the VM description. Policies for VM operations were documented as assumptions. For example:

1. A product can be dispensed if a credit is sufficient.
2. A product can be dispensed if the machine has sufficient stock.
3. A change can be produced if there is enough money in the machine's reserve.
4. Invariant: sum of products (in cash equivalent) and the reserve should be constant for a particular machine during its service time.

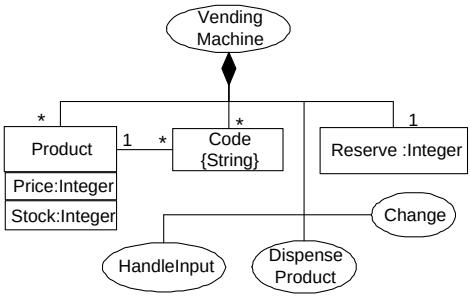

Fig. 4. IV Specification of VM. VM includes main concepts: Product, Code, and Reserve. Main actions (shown as ovals): HandleInput, DispenseProduct and Change.

In Fig. 5 we give the refined specification of the VM. New actions were added to explain the vending machine functionality. All the assumptions, made on the previous level, were transformed into actions with pre- and post- conditions. This specification can be easily mapped into the ASM notation (AsmL code in our case). Finally, an AsmL code was obtained[4] from the specification in Fig. 5. In this work the AsmL code was created by hand together with a rules for translation. Automatic AsmL code generation is a part of the future work.

For the Vending Machine we obtained the identical AsmL specification with the existed specification from the VM case study. As an illustration, below we propose an example of AsmL code for `SaleReady` operation (Fig. 5)

[3] In this work we simplified an original specification of VM [10]and reduced some requirements, such as restriction on the coin and bill denominations that the machine can accept and the ability to recognize coins and bills. Also in this example we assume that the customer always types the valid code of a product.

[4] AsmL Code for Vending Machine example is available at `http://lamspeople.epfl.ch/rychkova/Report%2008.07.2003/AsmLCode.pdf`

```
SaleReady() as Boolean
 return (Selection <> ['0','0']
         and not EmptyStock(ProductName())
         and Credit >= Price(ProductName())
         and not NoChange(Credit, Price(ProductName())))
```

Method `SaleReady` reflects the VM policies and returns "True" if no Null product is selected, selected product is available, credit is sufficient, and the machine has enough money to make change if necessary.

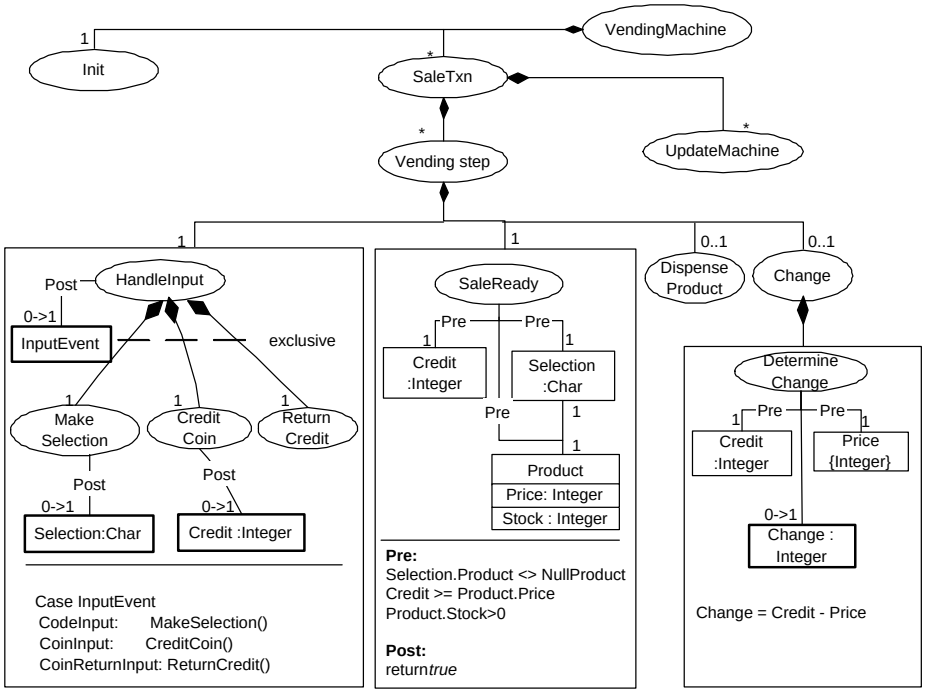

Fig. 5. Refined SEAM specification for VM. In this specification all necessary concepts, operations, and order of operations are shown. Necessary system policies formulated as a pre- and postconditions. AsmL code can be generated for each VM operation. Model can be read as following: Vending machine performance includes one `Init` operation and then set of `SaleTxn` (sale transaction) operations, where sale transaction can be considered as a dialog with one customer. Each sale transaction consists of a set of Vending steps (machine reaction on one input event) and may be finished by `UpdateMachine` operation. Each vending step can be described as a sequence of `HendleInput`, `SaleReady`, `DispenseProduct`, and `Change` operations.

As a result of this work, several basic rules for a mapping between the SEAM and ASM notations (namely, SEAM to AsmL translation) were generated.

5 Conclusion

We have presented an enterprise architecture (EA), an approach that allows a multi-disciplinary team to design enterprises (i.e. organizations and IT systems). To model complex systems with an EA team, it is important to have a unified modeling notation. For this purpose we propose the SEAM. However, to allow modelers to work more effectively, it is also important that models can be simulated and checked. This is a reason why we work on a mapping between our SEAM modeling language and ASM (for which simulation and model checking does exist).

We can highlight, that Egon Börger and Robert Stärk mention in [5] that ASM needs a graphical language that provides a "data model together with a functional model" to be usable with actual customers. With SEAM we provide such a notation.

References

1. Wegmann, A.: On the systemic enterprise architecture methodology (SEAM). Published at the International Conference on Enterprise Information Systems 2003 (ICEIS 2003), Angers, France.
2. Wegmann, A., Preiss, O.: MDA in Enterprise Architecture? The Living System Theory on the Rescue... Published at the 7th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2003) September 16-19 2003, Brisbane, Australia.
3. Naumenko, A., Wegmann, A., Atkinson, C.: The Role of Tarski's Declarative Semantics in the Design of Modeling Languages. Technical report.
4. Dori, D.: Object-Process Methodology - A Holistic Systems Paradigm. Springer Verlag, Berlin Heidelberg New York (2000)
5. Börger, E., Stärk, R.: Abstract State Machines. A Method for High-Level System Design and Analysis. Springer-Verlag, Berlin Heidelberg New York (2003)
6. Weinberg, M. W.: An Introduction to General System Thinking. Dorset House (2001).
7. Reference model of open distributed processing part 1. Draft International Standard (DIS), Helsinki, Finland, (15-18 May 1995)
8. Del Castillo, G.: The ASM Workbench. A tool environment for computer-aided analisis and validation of Abstract State Machine models. Dissertation. Fachbereich Matematik / Informatik und Heinz Nixdorf Institut Universitt Paderborn. Paderborn, 2000.
9. AsmL: The Abstract State Machine Language. Documentation prepared for Microsoft Research by Modeled Computation LLC, (2002) `http://www.modeled-computation.com`
10. Introducing AsmL: A tutorial for the Abstract State Machine Language. Vending Machine Case study. 2001, 2002 Microsoft Corporation. December 2001. `http://research.micosoft.com/foundations/AsmL`
11. D'Souza, D., Wills, A.C.: Objects, Components, and Frameworks with UML. The Catalysis Approach. Addison Wesley Longman, Inc. (1999)