Ontology for SEAM Service Models

Gorica Tapandjieva and Alain Wegmann
LAMS, IC, EPFL, 1015 Lausanne, Switzerland
{*gorica.tapandjieva, alain.wegmann*}*@epfl.ch*

Keywords: SEAM, Service Modeling, Ontology, Meta-model, Alloy, Formalization

Abstract: A service system is a popular concept in academia and industry. At the same time, it is a challenging concept to represent, due to its recursive nature and difficulty to relate it to entities in reality. In this paper we present an ontology for modeling service systems using the SEAM systemic method. Our ontology represents an updated and minimalistic version of the existing SEAM service modeling language that puts an emphasis on the behavior. The research approach we used is the design science for information systems research and it resulted with the ontology artifact. As part of the ontology, we provide a meta-model, well-formedness rules and formalization in the Alloy language. We conclude with presenting the limitations and a brief discussion on the contribution of shifting the focus towards the behavior in service systems.

1 INTRODUCTION

Services are a powerful abstraction of the value exchange and value creation within and across systems. The first foundational premise of S-D logic states that a “service is the fundamental basis of exchange” (Vargo and Lusch, 2008). Hence, the interaction between a company (service provider) and a customer (service consumer) is conceptualized as a service exchange. But also within a company there exist exchanges of services between an internal service provider and an internal consumer. For example, a company’s finances department manages the monthly salary payment for all employees. The service of salary payment is provided by the the finances department and it is consumed by all employees.

The entities that exchange services are known as service systems and they are defined as “a value-coproduction configuration of people, technology, other internal and external *service systems*, and shared information (such as language, processes, metrics, prices, policies, and laws)” (Spohrer et al., 2007) (italics added). This definition reveals the recursive nature of service systems, meaning that service systems are part of both the internal and external structure of one service system.

With their recursive nature, services are used and developed in different domains. Accordingly, service modelers from different functional perspectives need a generic, scalable, but yet rigorous modeling language for services. Such a language should enable service modelers to communicate their perception of the services recursion in a model and to show the structure of one service system.

To answer to the modelers needs, LAMS (LAMS, nd) researchers have been developing the SEAM[1] language (Wegmann, 2003; Systemic Modeling Laboratory LAMS, nd) that, among other things, is used in services modeling. As suggested by S-D logic and service science, SEAM (1) considers services as the fundamental basis of exchange and (2) models the observed reality as a recursive hierarchy of service systems.

1.1 Context and Motivation

A modeling language, including SEAM, conforms to a modeling ontology. The online dictionary Merriam-Webster[2] defines ontology as:
(1) A branch of metaphysics concerned with the nature and relations of being. Ontology deals with abstract entities.
(2) A particular theory about the nature of being or the kinds of things that have existence.

We do not provide philosophical arguments about the things that have existence, our scope are only things that are perceived and then modeled. Hence, we adopt the computer science definition of an ontol-

[1]SEAM is a family of methods used for consulting and teaching strategic thinking, business/IT alignment, and requirements engineering.

[2]`https://www.merriam-webster.com/dictionary/ontology`

Camera-ready copy in preparation.
To be presented at the 20th International Conference on Enterprise Information Systems (ICEIS), March 21-24, 2018 in Funchal, Madeira, Portugal

ogy – "a set of representational primitives with which to model a domain of knowledge or discourse" (Gruber, 2009). As system thinkers, when using SEAM, the entities we choose to perceive are systems, where: *A service is the behavior of a system, observed from the system's environment, that brings value to another system in the same environment.* As a consequence, all perceived systems are service systems and we use these two terms interchangeably.

The inspiration for our work came from our collaboration with our university's IT department, where we were involved in modeling existing and new services. In that project we noticed that the constructs used in a model have labels (names) that relate to the things observed and conceptualized in the reality. But service systems often do not relate to a predefined entity. For example, does an unofficial collaboration between two departments that provide services has a name? After making the collaboration official, what will influence the name? This brought us to the research question of this paper: *What influences a service system's name (label) after being conceptualized?*

We answer this question by designing a minimalistic ontology of the SEAM service modeling language that puts an emphasis on the behavior of service systems. The contribution of our ontology is that service modelers are encouraged to consider the system structure, and the name, as being emergent from the perceived or desired behavior of the service system. Note that our goal was not to develop a universal ontology of what exists, but to develop an ontology that can be used to design and describe services in a model.

In the remaining of the paper, in Section 2 we present the related work in modeling ontologies. Then, in Section 3, we present the service modeling with SEAM and we give an example. The instance of the research method we follow for designing the ontology is described in Section 4. In Section 5, we formalize the constructs used to build SEAM service models (meta-model and well-formedness rules). The evaluation is described in Subsection 5.4. We finish this paper by concluding in Section 6.

2 RELATED WORK

For services modeling, many different perspectives can be adopted. A service modeling language is adapted not only to the perspective, but to the motivation and the needs for representing the information around services. In this section we give an overview of the modeling languages in what we find to be two major services perspectives: computer science and marketing/management.

Computer Science Perspective

The computer science perspective of services is widely known as service-oriented architecture (SOA) and mostly applies to the usage of software solutions to facilitate the interaction of value co-creation between providers and consumers. Consequently, SOA research and application focuses on technical architecture that orchestrates software services, such as WS-* web services, APIs and RESTful services, in heterogeneous and distributed environments. As a service approach, SOA tries to separate the concern between the service description and implementation (Arsanjani, 2004). In an SOA context, services are loosely coupled, platform-independent, abstract the implementation and enable interoperability among systems. This enables services to "be combined and used by business processes that may span multiple service providers and organizations" (Georgakopoulos and Papazoglou, 2008).

The Service Oriented Architecture Modeling Language (SoaML) (SoaML v1.0.1, 2012) is specification project from the Object Management Group (OMG) (OMG, na) that provides a standard way to design, architect and model services within an SOA. The SoaML specification describes (1) a metamodel and (2) a set of extensions to the basic UML model elements, called a UML profile. Since SoaML is based on UML, it can be used with existing UML modeling tools. Besides services architectures, showing how services are implemented and used, SoaML models show the encapsulation of interactions between service participants (Amsden, 2014). Modeling with SoaML fits the model-driven development approach.

The Service Modeling Language (SML) (Popescu et al., 2009) and the Web Service Description Language (WSDL) (Christensen et al., 2001), are XML based modeling languages. The XML files describing the service contain information about the service configuration, deployment, monitoring, etc. These kind of modeling languages are not graphical, so they are not used in people's communication, but they are suitable for task automation, implementing interoperability, communication and exchange between applications, etc. The Unified Service Description Language (USDL) (Cardoso et al., 2010) is also based on XML that aims at unifying the technical and the business perspectives of a given service.

The Web Services Business Process Execution Language (WS-BPEL, only BPEL for short) (Rosen et al., 2008) is an XML-based language used to define the coordination and integration of Web Services within higher-level business processes of a company.

BPEL is platform independent and provides independence and flexibility by allowing the separation of the business process interaction from the web services (Rosen et al., 2008).

Enterprise Architecture (EA) is another discipline where services are modeled. We give a short overview of ArchiMate®, a widely adopted EA modeling language. As with any EA modeling language, ArchiMate aids the technology-integration efforts by creating models that within and between different domains. The models are used for visualization and analysis. ArchiMate expresses service-orientation with the so-called service layers and service implementation layers that realize the services. Services from a higher layer are linked with and typically use services from the lower layers. ArchiMate identifies three main layers: business, application and technology (Lankhorst, 2009), but the most recent specification includes a strategy and a physical layer (Josey et al., 2016). The concepts used for modeling in each layer have three dimensions: structural, behavioral and internal/external (Lankhorst, 2009, pg. 89).

Marketing/Management Perspective

Even before the introduction of SOA, services existed in the marketing discipline (Hill, 1977) with the focus on the service consumers, such as customers, users and clients. In our context, the marketing/-management perspective of services always takes into consideration the value that might arise from using information systems (IS) and IT in the service customization, and uses modeling languages that express service offerings, without over-analyzing the technical implementation. The focus is mostly on understanding the needs of consumers and designing service offerings consumers will value.

The visual representation of the company's processes that are exposed as business services is done with Business Process Modeling Notation (BPMN) diagrams (Rosen et al., 2008). BPMN has a notation that is understandable by business analysts, managers and technical developers. There is a possibility to compile BPMN diagrams into executable BPEL, and (White, 2005) has demonstrated how. This makes BPMN and BPEL the bridge between the computer science and marketing/management perspectives.

The e3service is an approach for generating service bundles, by using the notion of functional consequence to match the customer perspective and the supplier perspective (Razo-Zapata et al., 2015). The approach includes an ontology of constructs for service marketing and belongs to the e3family of ontologies for building service networks (Razo-Zapata et al., 2012). Automated reasoning can be used for finding service bundles that match the customer needs with the service outcomes from the supplier perspective (Razo-Zapata et al., 2015).

Business models are used for the analysis and design of the business logic of a company (Osterwalder, 2004). There are several service approaches that have been inspired by the widely cited and broadly applied Business Model Canvas (BMC) (Osterwalder and Pigneur, 2010). The Service Logic Business Model Canvas (Ojasalo and Ojasalo, 2015) includes the original 9 blocks from the BMC and considers a provider viewpoint ("From our point of view") and a customer viewpoint ("From customer point of view") in each of the blocks. In this manner, (Ojasalo and Ojasalo, 2015) incorporate the service logic into the business logic principles of BMC. Another approach is the Service Business Model Canvas (SBMC) (Zolnowski et al., 2014) that modifies the BMC to represent co-creation in the model. Similarly to the SLBMC, the SBMC adds a customer perspective. Finally, there is another canvas visualization approach, the Service Model Canvas (SMC) (Turner, 2015a; Turner, 2015b), designed by a user-experience professional. The SMC is intended to be used in an early exploration of a service by asking starter questions to organize and document service design thoughts.

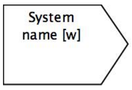
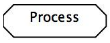
SEAM is another modeling method, coming with its own language, that is used for services modeling. It enables modelers to explicitly show different viewpoints of an organization. Over 15 years of research and practical application of SEAM have resulted with modeling principles, heuristics and constructs for representing and analyzing different abstraction levels of systems and behavior. In the next section we present details of the SEAM modeling technique on a concrete example.

3 INTRODUCTION TO SERVICE MODELING WITH SEAM

In a nutshell, SEAM represents an organization as a *hierarchy of systems* (from business down to IT) that provide *services*, where a system refers to entities in the reality perceived: a department, an employee, an IT system, or an application (Wegmann et al., 2008).

In the mentioned hierarchy, a system can have two views: abstract and concrete. To show the abstract view, a system is modeled as a whole, denoted with [w], in which the systems components are ignored and the focus is on the behavior provided as a service. In the concrete view, a system is modeled as a composite in which other systems as a whole are displayed with the relationships among these systems' services. A brief overview of the SEAM modeling constructs

Table 1: SEAM Modeling language visual vocabulary

System	
	*System* – an entity in the perceived reality, such as a company, a department, an organization. A system has two views, whole and composite. A small letter in square brackets, [w] or [c], is added as a suffix to the system name to denote which view is shown.
System behavior (Wegmann et al., 2008)	
	*Service* – the behavior of a system as a whole representing *a service offered by a system*.
	*Process* – the behavior of a system as a composite defining *a service implementation*.
Links	
	Use (invoke) link between services and processes, in a system as a composite. This link means that the process uses (invokes) the connected services.
	Refinement (decomposition) link, connecting the abstract and concrete view of a system, [w] and [c] respectively. The services from the system as a whole have corresponding processes in the system as a composite. These processes show the implementation of services.

used in this paper is presented in Table 1.

In the next part, we present the Infoscience example where we apply SEAM concepts for creating a two-level service model. The purpose of this model is to illustrate the SEAM modeling process on a real scenario. Note that to keep the model simple, we omit many details.

Example: A University's Infoscience Tool

Imagine reading a university's annual report in which there is a section about the research performance in terms of scientific publications. How does the university's management measure such performance? The answer lies in Infoscience, a tool that enables the management and access to scientific outputs. This tool is the result of a partnership between the librarians and the IT department. The main users of Infoscience are the university's researchers, who utilize it in the process of archiving their scientific production, organizing it, and afterwards, if needed, reclaiming the content. Besides researchers, Infoscience is used by the university's management and deans offices in the analysis of publications and citations.

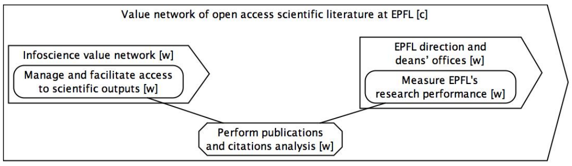


Figure 1: An example of a SEAM service model. The model shows the service of management and access to scientific outputs, as it is used by the university's direction and deans offices.

In SEAM, such a scenario is conceptualized as following. Every actor in the case description corresponds to a service system. We do not know the details of the partnership around Infoscience, so it is a black box (system as a whole), that we name *Infoscience value network*. In this case, the name is the choice of the modeler because no such department or organization (Infoscience value network) exists. The behavior of this partnership is the service that we call *Manage and facilitate access to scientific outputs*. Similarly, we conceptualize *University's direction and deans offices*, existing entities, as a black box, with the behavior *Measure the university's research performance*. These two systems combine their behavior in a the process we conceptualize as *Perform publications and citations analysis*. They are also the components of the composite system (the white box) that we name *Value network of open access scientific literature at a university*. Again, this value network is not recognized as an official entity. The resulting model is depicted in Figure 1.

We use SEAM to conceptualize the reality into models that are useful. Creating a SEAM model for the Infoscience tool is helpful to the university's top management for understanding how and by who are publication metrics gathered, processed. As we are presenting the SEAM modeling process, we will omit many scenarios and we will develop a generic model.

Who belongs to the Infoscience value network?

- A **steering committee** which consisted of high level stakeholders and experts like the dean of research, the head of the library, the head of an IT unit, the IT systems coordinator, etc. This committee is responsible for *making strategic decisions* concerning the Infoscience project.
- **Representative librarians** for the university sections and faculties, responsible for providing *support* to the researchers of the corresponding section.
- The Infoscience **technical coordinator**, responsible for the *day to day operations, third level user*

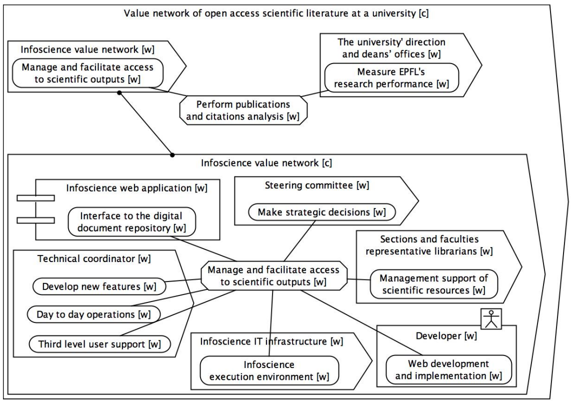


Figure 2: SEAM model with two organizational levels for the Infoscience project

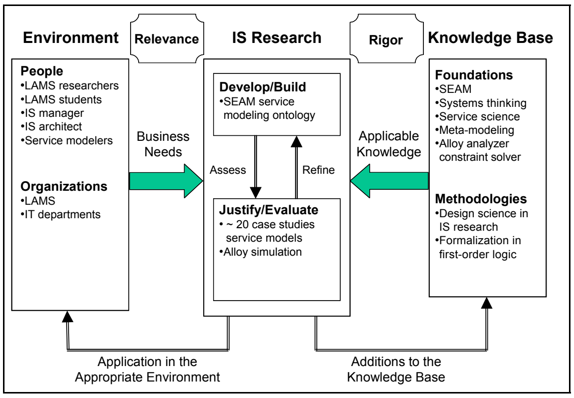


Figure 3: The IS research framework used for building the SEAM ontology artifact. The figure is adapted from (Hevner et al., 2004) to show the concrete environment, knowledge base and the research cycle.

support and development of new features.

- The Infoscience **developer**, responsible for the *web development and implementation of new Infoscience features.*
- The **Infoscience web application**, serving as a *web interface* to the digital document repository.
- A bundle of IT infrastructure resources and people providing the execution environment for the Infoscience application.

In the second level we show this *Infoscience value network* system as a composite with the process that implements the mentioned *Manage and facilitate access to scientific resources*. This process uses all the services from all of the composing systems seen as a whole that are part of the *Infoscience value network* system. There is one person, the *Developer*, and one application, the *Infoscience web application*, among these systems. In Figure 2, we illustrate the SEAM service model showing the interactions and the service exchange between the systems collaborating in the implementation of the *Manage and facilitate access to scientific outputs*.

In a service-oriented environment, there is no definite number of levels between two systems. We can systematically continue the modeling in the lower levels. For example, the *Infoscience IT infrastructure* system can be expanded to show the service implementation and the composing systems, most of them IT systems. In some modeling approaches, like ArchiMate (Josey et al., 2016), there is only one application layer where all applications must be modeled, so our example with IT systems at the second (*Infoscience value network*) and third level (*Infoscience IT infrastructure*) would not be supported. As a consequence, we need a meta-model that allows for scalability in terms of levels. Hence, a taxonomy for the levels is not present in SEAM service models, but modelers are free to embed a taxonomy in their diagrams, as one can do with our SEAM models, and say that there is an end user level (Figure 1), and several application levels.

4 RESEARCH METHOD: DESIGN SCIENCE IN INFORMATION SYSTEMS RESEARCH

The research method we used in the development of the SEAM service modeling ontology conforms to the design science in information systems research framework proposed by (Hevner et al., 2004). In Figure 3 we depict an instance of this framework to present the environment driving our research cycle and the knowledge base we used. All our activities followed the guidelines presented by (Hevner et al., 2004, p. 82) which have assisted us in understanding the requirements for an effective design-science research. Next, we elaborate how we fulfilled the recommendations in each guideline.

Guideline 1: Design as an Artifact

Our research resulted with the SEAM service modeling ontology artifact, which represents an updated, flexible, abstract, but yet easy to use in constructing service models. The artifact releases the existing constraint of not having a cycle between hierarchical levels and it emphasizes the behavior in service models

Guideline 2: Problem Relevance

The business needs we address with our research come from people that belong to two broad categories: academics and practitioners.

- *Academics* – LAMS researchers and LAMS students, are in a constant search of tools and methods that solve problems practitioners have in different domains. Their approach is always based on SEAM by applying a systems thinking perspective with a service-oriented solution. The simplified SEAM service modeling ontology helps LAMS researchers to apply the SEAM method with ease in projects that span across multiple domains. It also helps them to define the service system as an existing one or as a collaboration between actors that strive towards the same behavior.

- *Practitioners* – Our research is based on a collaboration with practitioners from our university's IT department, namely an IS manager and an IS architect. We study their perspectives and try to address their concrete needs:
  - The *IS manager*, has the concerns of how services are provided to the external customers and which entities are involved in the service implementation.
  - The *IS architect*, has the concerns of how to organize service architectures and how architectural designs can be effectively shared among all IT employees, including the IS managers.

  Both of these people have been using the SEAM modeling language for internal communication, but before our artifact they were never exposed to the SEAM meta-model. We noticed they sometimes they were as well struggling to define names for collaborating systems.

Guideline 3: Design Evaluation
We used two techniques to evaluate and justify our artifact.

1. *Case studies*: Through the course of our research, we have interviewed and collected information on around 20 case studies. Each of the cases involved solving a different problem that included service systems where our artifact was used in conceptualizing and modeling the problem situation and solution. In Section 3 we show only one of them.
2. *Formal model simulation*: We formalized our artifact in a first-order logic language (Alloy) and simulated it with the Alloy analyzer to analyze and derive modeling rules.

Guideline 4: Research Contributions
The main contribution of our artifact is the shift from focusing on systems structure towards focusing on the system behavior while modeling in SEAM. In the knowledge base, service science and SEAM benefit from a simplistic ontology applicable in all organizational levels. Both domains benefit from the reflection about naming service systems.

Guideline 5: Research Rigor
Our artifact is based on and conforms to the *SEAM systemic paradigm* (Wegmann, 2003). Consequently, we applied theories and concepts coming from *systems thinking* and *service science*. To be able to design and evaluate the ontology artifact, we used *meta-modeling* and the *Alloy constraint solver* tool.

Guideline 6: Design as a Search Process
Our search for an effective artifact had a main constraint of being capable to create "standard" SEAM models. To have such an artifact, we were building and evaluating our meta-model with the LAMS researchers and we included modeling rules coming from the standard SEAM.

Guideline 7: Communication of Research
The initial artifact has been communicated to the academic community via a publication and a poster presentation at a conference, (Tapandjieva and Wegmann, 2014), as a meta-model for automatic model generation. Technology-oriented audiences, namely IS architects, have read informal documentation and description of the artifact. We have not communicated the artifact to the management-oriented audiences in the form in which it is presented in this paper. According to our understanding, management-oriented audiences benefit from the visual model that represents an abstraction of their universe of discourse, so they do not need an additional abstraction in the form of an ontology or meta-model.

5 ARTIFACT: AN ONTOLOGY FOR SEAM SERVICE MODELS

In this section, we present the SEAM modeling language ontology in a more formal way. We first present the meta-model with the constructs that comprise the ontology for SEAM service models (Subsection 5.1). Then, we list the well-formedness rules that are not captured in the meta-model (Subsection 5.2). Afterwards, we formalize both the meta-model and the rules in a declarative language called Alloy (Jackson, 2002) (Subsection 5.3). Alloy generates instances of the meta-model to check if we have over or under-constrained the meta-model with our rules. With Alloy we verify the correctness of the meta-model and the modeling rules.

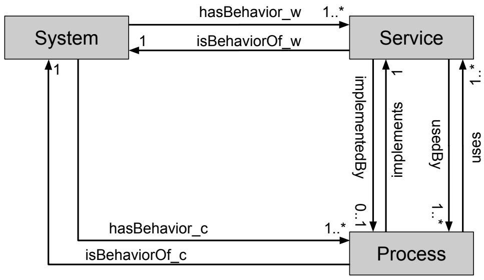


Figure 4: Meta-model of SEAM service modeling concepts used in this paper

5.1 SEAM Meta-model

Since 2008, researchers have been developing meta-models[3] that cover a bigger set of SEAM modeling constructs (Rychkova, 2008; Lê and Wegmann, 2013). In this paper we use SEAM models that focus on collaboration among services from different systems, namely people, IT systems, organizations, companies. Our interest is the value creation through such collaboration, from which new services emerge. Then, we study the surrounding context (upper level) where the new service is used, again in collaboration with other systems. And then again, understand the upper context, until we have interest in doing so. The example in Section 3 illustrates how we model collaborations with SEAM. In this paper we do not use all concepts defined in the existing meta-models, but we use only systems (whole and composite), behavior (service and process) and connections (decomposition and usage). We have built an ontology with the entities: *System, Service* and *Process*. Our proposed meta-model for this ontology is depicted in Figure 4. Every relationship between entities has a symmetric pair. We consider the cardinality of the starting entity to be always one. For example, the *hasBehaviorOf_w* one-to-many relationship between *System – Service* means that a system perceived as a whole can have one or more services, and the one-to-one *isBehaviorOf_w* relationship between *Service – System* means that one service can belong to only one system.

The concept of a system as a whole is captured in the *hasBehaviorOf_w* relationship, with services being the behavior of systems as a whole. Similarly the *hasBehaviorOf_c* means that the behavior of systems as a composite are processes.

The two kinds of links from Table 1 are captured in the relationships between *Process – Service*. The *implementedBy* and *implements* stands for the refinement and implementation concept. The usage (invoke) link is captured with the *uses* relationship, telling that a process can use multiple services. The *usedBy* relationship means that the same service can collaborate in different processes.

In the meta-model, we omit entities for the concepts of a system as a whole and system as a composite. As mentioned before, these concepts are represented with the relationships *behaviorOf_w* and *behaviorOf_c*. We also do not include a *System – System* relationship to denote that a system as a composite contains processes and systems as a whole. In short, the systems as a whole that belong to the system as a composite must participate with their services in the process; they are not free-floating in the composite. Consequently, the following query *System – behaviorOf_c – Process – uses – Service – behaviorOf_w – System* gives all systems as a whole belonging to the starting system as a composite. Additional details are presented in the next Subsection 5.2, with the modeling rules.

From a systems thinking perspective, the distinction between the concepts of a system as a whole and a composite is based in epistemology; these concepts depend on the observer's (the modeler's) knowledge and relation to the reality while describing the systems. It is the observer who decides the viewpoint he takes when modeling the reality: the context that he knows (system as a composite) where many systems as a whole interact (services connected to a process). In addition, for newly created services, often based on collaboration, there is no formal entity that hosts the process execution. In such cases, the observer derives the system name from the process (behavior). To conclude, we have two reasons for omitting the concepts of a whole and a composite from the meta-model, the first one being pragmatic (they can be computed), and the second philosophical.

5.2 Well-formedness Rules

The well-formedness rules of a modeling language complement the meta-model to ensure the consistency of models. They are also created to avoid the semantic ambiguities that might arise in the instances of the meta-model. We summarize the SEAM well-formedness rules in Table 2.

5.3 Formalization in Alloy

We use Alloy to formalize the SEAM meta-model concepts and the well-formedness rules. Then, we use the Alloy Analyzer to generate instance models or counter-examples in a domain we have specified. Getting meaningful instances would indicate that our

[3] In this paper we use the term meta-model and information model interchangeably.

Table 2: Well-formedness rules for SEAM service modeling

Rule	Description	Applies to concepts
R1	A service is unique to one system as a whole, so two systems as a whole cannot contain the same service.	*Service*
R2	A process is unique to one system as a composite, so two systems as a composite cannot contain the same process.	*Process*
R3	A process can implement only one service.	*Process* and *Service*
R4	A process must be connected to at least one service. Otherwise, there is no relationship among systems as a whole, and the overall observation of the system as a composite is put in question.	*Process* and *Service*
R5	The decomposition relationship, shows the refinement from the abstract view of a system (the whole) to the concrete view of a system (the composite). The service present in the whole, becomes a process in the composite view. As a consequence, the process, and the service it implements, must belong to the same system, so a process cannot implement a service from another system as a whole, different from the process' system as a composite.	*Process* and *Service*
R6	Recursion between levels is allowed. A process can be connected with the service it implements. Such recursion does not have to be immediate, it can happen at any level in the organizational hierarchy.	*Process* and *Service*

formalization is consistent in the domain we have set.

Alloy (Alloy, nd; Jackson, 2002) is a declarative structural modeling language based on first-order logic. It comes with a constraint solver, the Alloy Analyzer, that automatically finds models that satisfy the formulas written with the Alloy language. The Alloy code usually describes basic structures, called signatures, and has detailed constraints applied to the structures. Constraints are expressed in terms of facts, predicates, assertions or quantifiers.

The following code shows the complete formalization of the SEAM meta-model and the well-formedness rules. There is a signature (*sig* keyword) for each concept from the meta-model: *System, Service* and *Process*. The cardinalities are coded with the corresponding keywords: 1..* is mapped to *some*, 0..1 is mapped to *lone* and 1 is mapped to one.

```
sig System {
hasBehavior_w: some Service,
hasBehavior_c: set Process
}
sig Service {
isBehaviorOf_w: one System,
implementedBy: lone Process,
usedBy: set Process
}
sig Process {
isBehaviorOf_c: one System,
implements: one Service,                    //R3
uses: some Service                          //R4
}

fact uniqueServiceInSystem {                //R1
no ser: Service, s1: System, s2: System |
ser in s1.hasBehavior_w and
ser in s2.hasBehavior_w and s1!=s2
}
fact uniqueProcessInSystem {                //R2
no p: Process, s1: System, s2: System |
p in s1.hasBehavior_c and
p in s2.hasBehavior_c and s1!=s2
}
fact refinement {                           //R5
all s: Service, p: Process | p.implements=s =>
s.isBehaviorOf_w=p.isBehaviorOf_c and
s.implementedBy=p
all p: Process, s: Service | s.implementedBy=p =>
s.isBehaviorOf_w=p.isBehaviorOf_c and
p.implements=s
}
fact symmetry {
all s: Service, p: Process | s in p.uses => p in s.usedBy
all s: Service, p: Process |
p.implements=s<=>s.implementedBy=p
all sys: System, ser: Service |
ser.isBehaviorOf_w = sys<=>ser in sys.hasBehavior_w
all sys: System, p:Process |
p.isBehaviorOf_c = sys<=>p in sys.hasBehavior_c
}
run {} for exactly 4 Service, exactly 2 Process,
exactly 3 System
```

Listing 1: Meta-model formalization in Alloy

The well-formedness rules R3 and R4 are already captured with the cardinalities. The remaining rules are written as facts. We finally specify the domain for which we want the Alloy Analyzer to generate an instance model (the *run* command). The existence of an instance means that the Alloy code is correct and not over-constrained for the domain we have set.

The Alloy instance in Figure 6 is one of the many that satisfies the constraints written for the well-formedness rules for the domain *exactly 4 Service, exactly 2 Process, exactly 3 System*. Note the immediate cycle that exists between Service3 and Process0. Such cycles are allowed to capture situations found in reality. We demonstrate this with an example.

Imagine the hosting service offered by a data center. Such a service is implemented by using a server room, racks, power supply and other technical resources. In addition, the data center uses a web site to display information about the status of the resources, for monitoring purposes. This website is hosted on a machine in the data center.

Figure 5: Alloy automatically generated version of the meta-model. Note the similarity with the meta-model in Figure 4.

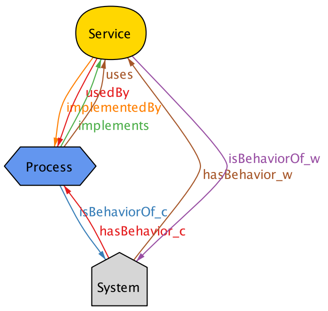
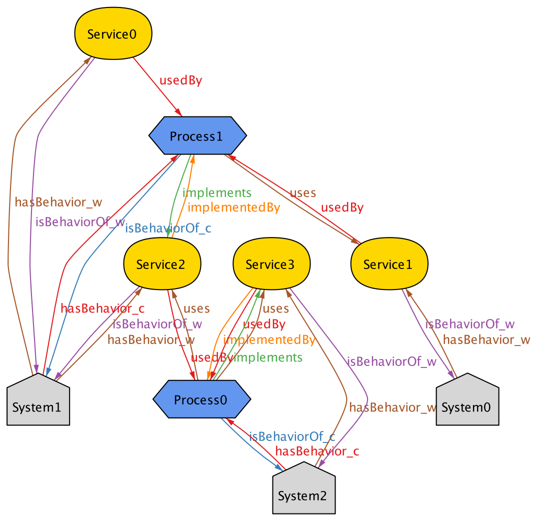


Figure 6: One Alloy-generated instance where all well-formedness rules a respected



5.4 Evaluation

The Alloy checking is a formal evaluation of the ontology. It proves that existing SEAM models, as the one showed in the Infoscience example can be generated with our meta-model. Another form of an evaluation are the around 20 case studies we modeled with our ontology. The difficulties we encountered in naming the systems were easily overcome by focusing on the behavior, i.e. the process and the emerging service. In addition, one IS manager and IS architect have been using our meta-model and have developed tools upon it. Their feedback has been positive. We still plan to conduct additional evaluation in the form of user studies, interviews with more practitioners and survey.

5.5 Limitations

For simplicity, many other aspects of SEAM are not presented and discussed in this paper. One is the functional hierarchy, more precisely the decomposition of the behavior into sub-services and sub-processes. In addition, the meta-model does not capture temporal and sequential elements, like change of state over time or after an action. This can be done by enriching the meta-model with entities that show properties of systems, services and processes. Previous SEAM meta-models included these concepts (Lê and Wegmann, 2013).

6 CONCLUSION

SEAM modeling focuses not only to the user perception of the value that the service brings, but it also gives details on the perception of the systems in the multiple organizational levels of the service provider. The service exchange and value co-creation among systems are present across the whole organization, not only at the end-user level. Every system, namely organization, person, IT application, provides a service, so with the proposed ontology we leverage on the general SEAM contribution of services being explicitly modeled at any organizational level of interest. Such conceptualization uses the first foundational premise of service-dominant logic: "service is the fundamental basis of exchange" (Vargo and Lusch, 2008).

SEAM had already been used in such contexts, but it provides a set of models and tools, that combined are difficult to use in a large scale projects. Their existing meta-models encompass entirely SEAM. In this paper we suggested to use a stripped down version of SEAM. For this version we propose an ontology that offers a new perspective, the focus on the behavior of the system, not the system itself. All the relationships among the *Service* and the *Process* concept in the meta-model shift the modeling focus around the behavior, i.e. what systems do together. In such a version, cycles are allowed and even external actors, such as regulators and suppliers, are considered in a service implementation.

Service science literature discusses mainly one level of value co-creation, the one with the customer or end user. Here, we propose to reuse the same principle inside an organization, by only focusing on what systems do together. With our ontology we do not introduce a classification of service providers and consumers. We focus on the interaction, collaboration and co-creation among systems, regardless of their position in the organizational hierarchy. The analysis of the dynamics between a service provider and consumer are systematically applicable on the systems in the service provider, internally, across the whole organization. We can leverage the existing knowledge on services, that applies at the consumer level, to organize a team, a department, even an entire organization or corporation.

During the course of our research, we have realized that the existence of systems (more precisely the systems we choose to observe) is strongly dependent on our perception of the behavior of these systems, i.e. services. From our involvement in industry projects we have learned that even the systems' names convey an information about the behavior of the system (a service or a process). In SEAM, processes show collaboration and value co-creation, so when we try do define the system with its boundaries where this collaboration happens, the choice for the system name (1) expresses the function of the collaboration, or (2) relates to an organization, department, or an entity from the observed reality. Overall, systems thinking is an epistemology (Mingers, 2006, p. 87). The concepts of a system as a whole and a systems as a composite only describe how we choose to perceive the reality, so they are not strictly represented in the meta-model of the ontology.

REFERENCES

Alloy (n.d.). Alloy: a language & tool for relational models. http://alloy.mit.edu/.

Amsden, J. (2014). Integrating BPMN and SoaML, part 1. motivation and approach. https://www.ibm.com/developerworks/rational/library/integrate-bpmn-soaml/index.html.

Arsanjani, A. (2004). Service-oriented modeling and architecture. https://www.ibm.com/developerworks/library/ws-soa-design1/.

Cardoso, J., Barros, A., May, N., and Kylau, U. (2010). Towards a unified service description language for the internet of services: Requirements and first developments. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 602–609. IEEE.

Christensen, E., Curbera, F., Meredith, G., Weerawarana, S., et al. (2001). Web services description language (wsdl) 1.1.

Georgakopoulos, D. and Papazoglou, M. P. (2008). Overview of service-oriented computing. In *Service-Oriented Computing*, chapter 1. The MIT Press.

Gruber, T. (2009). Ontology. http://tomgruber.org/writing/ontology-definition-2007.htm.

Hevner, A., March, S., Park, J., and Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1):75–105.

Hill, T. P. (1977). On goods and services. *Review of income and wealth*, 23(4):315–338.

Jackson, D. (2002). Alloy: A lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(2):256–290.

Josey, A., Lankhorst, M., Band, I., Jonkers, H., and Quartel, D. (2016). An introduction to the ArchiMate® 3.0 specification. *White Paper from The Open Group*.

LAMS (n.d.). Systemic modeling laboratory – LAMS, EPFL. http://lams.epfl.ch/.

Lankhorst, M. (2009). *Enterprise Architecture at Work: Modelling, Communication and Analysis*. Springer.

Lê, L.-S. and Wegmann, A. (2013). Hierarchy-oriented modeling of enterprise architecture using reference-model of open distributed processing. *Computer Standards & Interfaces*, 35(3):277–293.

Mingers, J. (2006). *Realising systems thinking: knowledge and action in management science*. Springer Science & Business Media.

Ojasalo, K. and Ojasalo, J. (2015). Adapting business model thinking to service logic: an empirical study on developing a service design tool. *THE NORDIC SCHOOL*, 309.

OMG (n.a.). About OMG. http://www.omg.org/about/.

Osterwalder, A. (2004). *The business model ontology: A proposition in a design science approach*. PhD thesis, L'Ecole des HEC de l'Université de Lausanne.

Osterwalder, A. and Pigneur, Y. (2010). *Business model generation: a handbook for visionaries, game changers, and challengers*. John Wiley & Sons.

Popescu, V., Pandit, B., and Smith, V. (2009). Service modeling language, version 1.1, W3C recommendation. https://www.w3.org/TR/sml/.

Razo-Zapata, I. S., De Leenheer, P., Gordijn, J., and Akkermans, H. (2012). Service network approaches. In *Handbook of service description*, chapter 3, pages 45–74. Springer.

Razo-Zapata, I. S., Gordijn, J., De Leenheer, P., and Wieringa, R. (2015). e3service: A critical reflection and future research. *Business & information systems engineering*, 57(1):51–59.

Rosen, M., Lublinsky, B., Smith, K. T., and Balcer, M. J. (2008). Applied soa: Service-oriented architecture and design strategies.

Rychkova, I. (2008). *Formal semantics for refinement verification of entreprise models*. PhD thesis, EPFL.

SoaML v1.0.1 (2012). OMG Service Oriented Architecture Modeling Language, Version 1.0.1. Technical report, Object Management Group (OMG).

Spohrer, J., Maglio, P. P., Bailey, J., and Gruhl, D. (2007). Steps toward a science of service systems. *Computer*, 40(1):71–77.

Systemic Modeling Laboratory LAMS, E. (n.d.). SEAM. http://lams.epfl.ch/seam/.

Tapandjieva, G. and Wegmann, A. (2014). Specification and implementation of a meta-model for information systems cartography. In *Proceedings of the CAiSE2014 Forum at the 26th International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 1164, pages 113–120. CEUR-WS.org.

Turner, N. (2015a). Introducing the service model canvas. http://www.uxforthemasses.com/service-model-canvas/.

Turner, N. (2015b). The updated service model canvas. http://www.uxforthemasses.com/updated-service-model-canvas/.

Vargo, S. L. and Lusch, R. F. (2008). Service-dominant logic: continuing the evolution. *Journal of the Academy of marketing Science*, 36(1):1–10.

Wegmann, A. (2003). On the Systemic Enterprise Architecture Methodology (SEAM). In *Published at the International Conference on Enterprise Information Systems 2003 (ICEIS 2003)*.

Wegmann, A., Kotsalainen, A., Matthey, L., Regev, G., and Giannattasio, A. (2008). 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.

White, S. (2005). Using BPMN to model a BPEL process.

Zolnowski, A., Weiß, C., and Bohmann, T. (2014). Representing service business models with the service business model canvas–the case of a mobile payment service in the retail industry. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 718–727. IEEE.