2016 Conference on Systems Engineering Research

Engaging ontologies to break MBSE tools boundaries through semantic mediation

Uri Shani[a], Shmuela Jacobs[b], Niva Wengrowicz[b], Dov Dori[c] *

[a]*IBM Haifa Research Lab, Haifa University, Haifa, Israel*
[b]*Faculty of Industrial Engineering and Management, Technion – Israel Institute of Technology, Haifa, Israel*
[c]*Engineering Systems Division, Massachusetts Institute of Technology,Cambridge MA, USA*

Abstract

We introduce ontologies (as in the Semantic Web ontology language OWL) to serve as formal description of the modeling languages of model-based systems engineering (MBSE) tools. In these tools, systems are designed via abstractions, each tool with its own conceptual modeling languages having different syntaxes and different semantics. This creates barriers in sharing these mathematical models among the tools. Our journey starts with the web, where large amounts of information from any sources on the Web can be linked and combined in many ways. The Semantic Web added meaning through OWL ontologies to the information in web pages so machines could better process information to enhance users' experience.

The field of MBSE offers a great many different tools to the engineer, each representing the system under design or a view of that system albeit mathematically, but differently so that sharing a model from one tool with another is impossible, or darn difficult and error prone.

This problem is often classified as "tools interoperability", but it is primarily a language interoperability problem. The Open Services for Lifecycle Collaboration (OSLC) specifications and initiative brought the Semantic Web technologies into MBSE by introducing Resource Description Framework (RDF) for common model representation, RESTful protocols, as a common communication and data exchange method, and enabling the linking of model elements in the different tools that are used for product lifecycle management (PLM)..

The semantic mediation container (SMC) is a platform developed by IBM Research as part of EU projects (SPRINT, DANSE and now PSYMBIOSYS) to support the mediation of models represented in RDF and exchanged over the Internet using RESTful API. SMC extends the benefits of the OSLC approach by adding semantics using the Web Ontology Language (OWL) specifications to define language ontologies. Each tool exchange models with the SMC platform, where models are constructed according to specific ontologies (in OWL) per each of the tools. Models can than be mediated to comply with different ontologies. The rules governing the mediation are also coded as OWL ontologies that bridge two different ontologies and are interpreted by a mediator. Bridging is a form of transformation, or an inference over the statements of the 3 involved ontologies, and those of the input RDF model, all driven by a mediation engine we term mediator. In this paper we introduce the first stages of mediation as applied to two different modeling tools; Rhapsody that implements the OMG standard SysML specification, and OPCAT – the OPM CASE tool implementing the Object-Model Methodology (OPM) which is an emerging ISO 19450 standard.

Keywords: MBSE; Semantic Web; ontologies; mediation; tools interoperability; OPM; SysML;

[1] Corresponding author. Tel.: +972-4-8296282; fax: +972-4-8296-116. *E-mail address:* shani@il.ibm.com

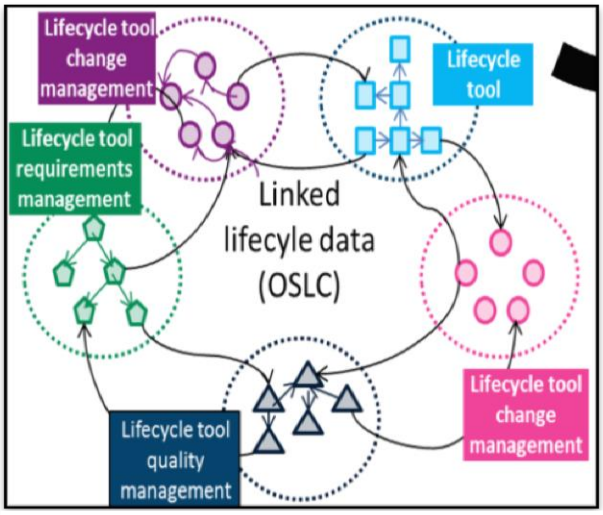
1. Introduction

As requirements from systems grow and technology advances, the systems we create become larger and more intricate. Whether it is software, a machine, or a combination of them, there are many aspects in different domains to be considered throughout the lifecycle of a system. Since systems engineering, addresses a large mix of domains in a single system or system-of-systems, it is difficult [1], if not impossible, to agree on a common, shared language. Specialized domains have established exchange potential among tools, e.g., the mechanical STEP (ISO 10303) [2] standard and the electronic EDIF (IEC 61690-1) [3].

Design, development, and manufacturing of many complex systems employs best of breed tools that span across different dimensions, including requirements, architecture, physical modeling (e.g., electrical and mechanical), software development, control systems design, various analytic tools (e.g. simulation, verification, etc.), and project management tools, which contain information about schedules, tasks, and costs. Moreover, vast amounts of experimental, product, and operational data exist and are often utilized during design process. Being able to leverage the knowledge of relationships, whether logical or mathematical, between all or some of the aforementioned information can bring great value to the system development process, reducing costs and risks, improving designs, and shortening schedules.

The Semantic Web is a relatively new approach for sharing data across the internet. It is managed and developed by the World Wide Web Consortium (W3C), which presents specifications and technologies that may be used for easier and more efficient collaboration between teams using different software tools. Among these specifications are Resource Description Framework (RDF) for data representation, Linked Data for specifying relations among resources, and Web Ontology Language (OWL) for ontologies definition. A most successful effort in applying Semantic Web principles into systems engineering is the Open Services for Lifecycle Collaboration (OSLC) initiative [4]. OSLC workgroups specify a common vocabulary that provides for interoperability between tools within and across different domains. A model may be instantiated to an existing, working system, the components and resources of which are uniquely identified by URIs and may be accessed by applications that use Representational State Transfer (RESTful [5]) protocol.

The idea of Linked Data [6,7] as applied in OSLC is depicted in Figure 1. Each tool holds information relevant to its domain, and Internet URLs (the black arrows) reference related addressable resources in other tools. For example, a requirement may address a test-case to test it, or correlating component architecture in a design diagram. OSLC requires MBSE tools to operate as web servers rather than traditional isolated tools, managing models. Model contents are exchanged with other tools and users via web service requests. However, most modeling tools are not working in this way, but are (thick) client tools, and they are not responsive web services.


Lifecycle tool change management
Lifecycle tool
Lifecycle tool requirements management
Linked lifecycle data (OSLC)
Lifecycle tool change management
Lifecycle tool quality management
Figure 1. Linking resources in different tools to create the OSLC tools interoperability.

The Semantic Mediation Container (SMC) is a container of services, originally built as a Jazz® [8] application, a platform which implements OSLC specifications with integrated tools. Presently, it is generalized to work with any web container and cloud platforms such as the IBM Bluemix®. In fact, SMC holds models of various domains of a system in RDF format, communicates them to and from client tools using RESTful protocols, and performs mediation among models, which is an key capability of SMC compared with “plain” OSLC providers of RDF models. Client tools can post (export) models serialized from their internal representation to RDF graph structure, and get (import) RDF models and merge them back with their internal structure. Once on the SMC server, RDF model resources (i.e., elements) can be accessed using the usual OSLC protocol. The mediation process is discussed in the following section. SMC has been developed in the SPRINT [9] and DANSE [10] EU FP7 projects for the collaborative design in complex systems engineering and for the design of systems of systems, respectively.

In this paper we introduce quite a few synonyms that are collected into the Nomenclature box shown here for one place to seek them all.

Nomenclature

OPCAT	OPM Case Tool
OPD	Object-Process Diagram
OPM	Object Process Methodology
OSLC	Open Services for Lifecycle Collaboration
OWL	Web Ontology Language
RDF	Resource Description Framework
REST	Representational State Transfer
SMC	Semantic Mediation Container
UPDM	Unified Profile for DODAF and MODAF
URI	Universal Resource Identifier
URL	Universal Resource Locator
W3C	World Wide Web Consortium

In the SPRINT project, three tools have been used to demonstrate a collaborative model development through multiple tools, each with its own specialty and purpose: Rhapsody® [11] for design using SysML[12], Wolfram SystemModeler® [13] for modeling and simulation using Modelica [14], and an in-house analysis tool called "DESYRE®". SPRINT use cases were also applied in the DANSE project with additional tools, increasing interoperability and adaptation to system of systems methodologies and standards like the UPDM [15] modeling framework, which has been used thorough the Rhapsody tool. The ontology for the Rhapsody tool has been developed with two purposes in mind: (1) allowing the mediation of models to the other tools in the project by including all the important concepts that are needed by the engineers, and (2) maintaining the RDF models human readable for manual verification of the mediation.

Object-Process Methodology (OPM) [16] has been the subject of research [17, 18] comparing it to SysML and investigating synergy between them [19], and applying it in a variety of system modeling domains, ranging from Executable Simulation Environment [20] through Project-Product Lifecycle Management [21, 22], risk modeling [23], systems engineering education [24], and conceptual model-based system biology [25].

As part of the research, OPM models are translated to SysML and vice-versa. The use of the semantic mediation approach to share OPM models with SysML tools such as Rhapsody allows easier implementation of the translation process, since Rhapsody is already integrated in SMC and is being evolved by its own development team. Hence, the following steps were needed to add OPM to SMC and allow translation to SysML: 1. Develop export module from OPM Case Tool (OPCAT[†], [26]) to RDF data structure, 2. Develop import module for RDF files into visual representation in OPCAT, and, 3. Define rules for mediation between OPM to SysML models. All three steps require the OPM ontology defined using OWL. Completion of this work shall result in more than merely the ability to mediate OPM and SysML models. Using existing transformations between SysML to other languages, model data and concepts may be shared between OPCAT and a large variety of modeling tools. Furthermore, having the ability to represent OPM models in RDF format is the first step for integrating OPM into the Semantic Web.

The rest of this paper comprises four sections: The SMC platform and the semantic mediation concept, the Rhapsody ontology development, the OPM ontology development, and Summary and future research.

2. Semantic Mediation Container

OSLC is concerned with linked data so that model elements in one tool can maintain links to elements in other models, regardless of the tool and language or semantics of the tools. Having a link to an element, OSLC supports the access to the element description as a resource with properties, structured as an RDF graph. To further browse a link over the Internet, the resource owner is designed as a Web server, which can also respond with structural information about the "shape" of the model graph in which the resource is located, and even the visual image diagram and icons of that resource. All these facilities make it possible for an engineer user to browse models owned by any OSLC compliant server. However, for machines to formally process the linked data, the semantics of the linked resources are needed. That is where ontologies play an important role. The ontologies are specified in OWL and are also represented as RDF graphs.

The semantic mediation container (SMC) assumes that all RDF model graphs are associated with an ontology that provides the semantics of the graph elements and thus enables mediating between a model with a certain

[†] OPCAT is available freely from http://esml.iem.technion.ac.il/

ontology and a model with a different ontology. In other words, the SMC transforms a model in one modeling language to a model in another. To better understand our terminology, in what follows we present simple example of two models, each expressed in its own ontology, which are mediated using simple mediation rules.

2.1. RDF model example

Figure 2 is of an RDF model, represented graphically on the right and textually on the left, using the RDF Turtle syntax [27]. The top *@prefix* lines define namespaces, so their prefix string can be used to shorthand the full name-space. The *uml* prefix is for the namespace of the ontology describing the language of this model (see subsection 2.2). *base* is the prefix for the namespace of the model elements. The bold face URLs are resources in this model, represented as nodes in the RDF graph. Each resource is followed by properties, called "predicates" in the RDF terminology, which are edges to other nodes in the graph. Some nodes are literals, some are resources in the same model graph, and some are nodes in other graphs, such as ontologies. The elements with the prefixes *uml*, *rdf*, and *rdfs* are resources in other models of ontologies. The RDF and RDFS ontologies provide the basic building blocks of RDF graphs, and the UML ontology is a new ontology, as described in the next subsection.

The four resources in this graph (…0002, …0003. …0004, and …0005) are "instances" of the two ontological concepts *uml:Type* and *uml:Object***.**

```
@prefix base:
<https://sprint.haifa.il.ibm.com:9444/dm/sm/repository/uml/resource/> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix uml:    <http://tutorial.uml/> .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .

base:0002      a      uml:Type ;
rdfs:description "Class A" ;
rdfs:title "A" ;
uml:hasObject  base:0003 .

base:0005      a      uml:Type ;
rdfs:description "Class B" ;
rdfs:title "B" ;
uml:hasObject base:0004 .

base:0004      a      uml:Object ;
rdfs:description "An instance of class A" ;
rdfs:title "aA" ;
uml:hasType base:0002.

base:0003      a      uml:Object ;
rdfs:title "aB" ;
uml:hasType  base:0005.
```

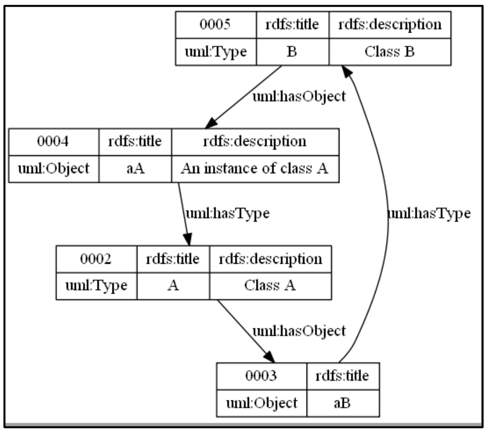


Figure 2. Turtle syntax (left) of a simple model RDF graph, and the diagram (right) of that graph.

```
@prefix uml:    <http://tutorial.uml/> .
@prefix rdf:    <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:    <http://www.w3.org/2002/07/owl#> .
@prefix rdfs:   <http://www.w3.org/2000/01/rdf-schema#> .
uml:     a      owl:Ontology  .
uml:hasObject      a      owl:ObjectProperty ;      rdfs:domain   uml:Type ;
      rdfs:range    uml:Object .
rdfs:label                  rdfs:domain    uml:TopUML .
rdfs:comment                rdfs:domain    uml:TopUML .
uml:hasType
      a      owl:ObjectProperty ;
      rdfs:domain               uml:Object ;
      rdfs:range                uml:Type .
uml:TopUML      a      owl:Class .
uml:Object       a      owl:Class ;      rdfs:subClassOf    uml:TopUML .
uml:Type         a      owl:Class ;      rdfs:subClassOf    uml:TopUML .
```

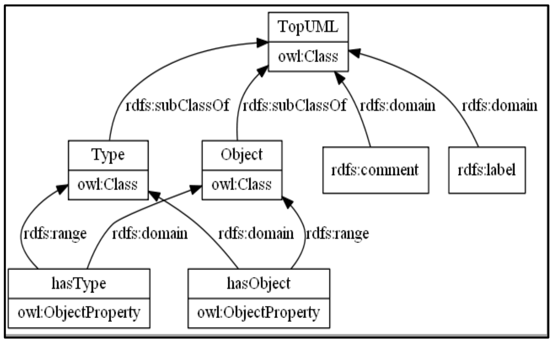


Figure 3. Ontology RDF graph on left, and a diagram of that graph on the right for the ontology "*UML*" used to define the model in Fig. 2.

2.2. Ontology

For the model of Figure 2, discussed above, we introduce in Figure 3 its ontology, which is an RDF graph as well. The graph defines resources which are concepts of this ontology. These are used to define the resources of the instance graph in Figure 2. The types of these resources are concepts of the OWL ontology, hence OWL is the meta-model of the various ontologies.

The ontology is an RDF graph as well. The prefix uml‡ represents the namespace of the ontology *http://tutorial.uml*. It is a resource of type *owl:Ontology*. The concepts *rdfs:label* and *rdfs:comment* are defined in the *rdfs* standard taxonomy, but here they are also refined as having a relation *rdfs:domain* with a resource concept of this new ontology. The main concepts are the resources *uml:Type* and *uml:Object* which are OWL classes having type *owl:Class*. The main predicate concepts are *uml:hasType*, and *uml:hasObject*, which are OWL properties, as expressed by their type *owl:objectProperty*. Other relations and resources in this ontology are used to assign the *rdfs:title* as the name of an entity and *rdfs:comment* as its textual description (see Figure 2).

2.3. Mediation rules

The mediation bridges over two ontologies. For the purpose of this example, we create a second ontology, SML (having namespace *http://tutorial.sml/*) for which we define a prefix *sml:*), which defines the concepts *sml:Block*, *sml:Part*, *sml:hasPart*, and *sml:hasBlock*. This ontology is similar to the “UML” one, as it looks like a mere renaming of concepts, which serves well this introduction. Therefore, it is simple to mediate between the two. The mediation rules are defined as a bridging ontology, presented in Figure 4.

When applied to the UML instance model in Figure 2, it is mediated to the model on the right hand side of Figure 4 by redefining the resources, reclassifying them with the corresponding classes of the SML ontology, and converting the relation properties to the corresponding relations in that SML ontology.

```
@prefix rules:    <http://tutorial.sml.uml/> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .

<http://tutorial.sml/hasBlock>
     owl:equivalentProperty
           <http://tutorial.uml/hasType> .

<http://tutorial.sml/Block>
     owl:equivalentClass <http://tutorial.uml/Type> .

<http://tutorial.sml/hasPart>
     owl:equivalentProperty
           <http://tutorial.uml/hasObject> .

<http://tutorial.sml/Part>
     owl:equivalentClass <http://tutorial.uml/Object> .

rules:
     a      owl:Ontology ;
     owl:imports <http://tutorial.sml/> , <http://tutorial.uml/> ;
     owl:versionIRI <http://tutorial.sml.uml/v1.0> .
```

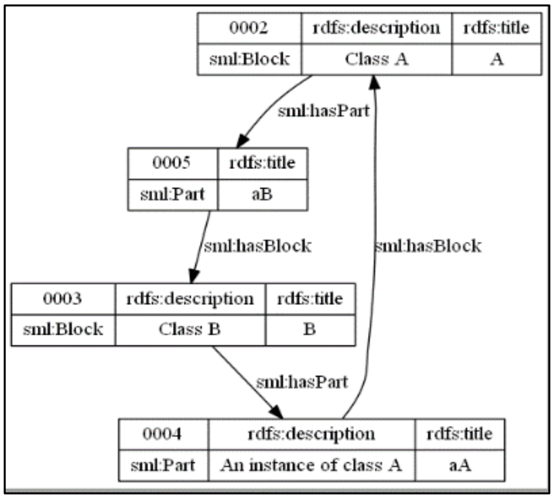


Figure 4. The rules ontology (left) in Turtle format, and the resulting model instance of the AML ontology, mediated from the UML ontology model instance on the right.

‡ This *uml* term should not be confused with the UML standard, although the concepts here have similar meaning to corresponding concepts in UML.

2.4. *The SMC platform*

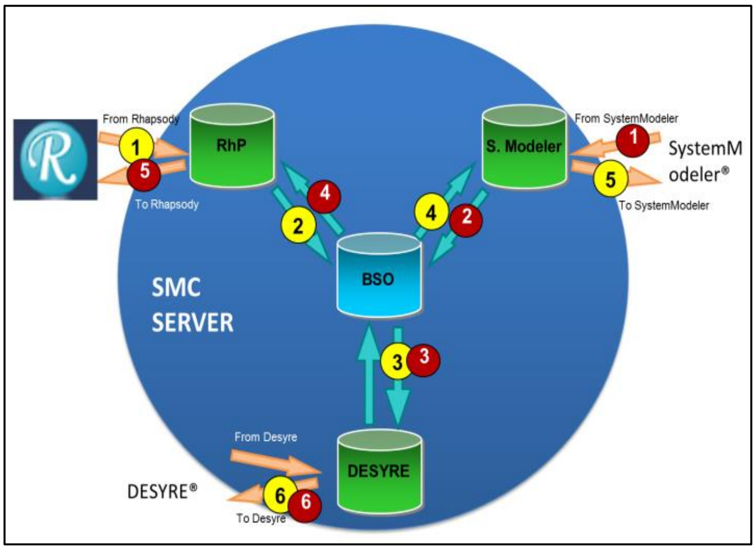
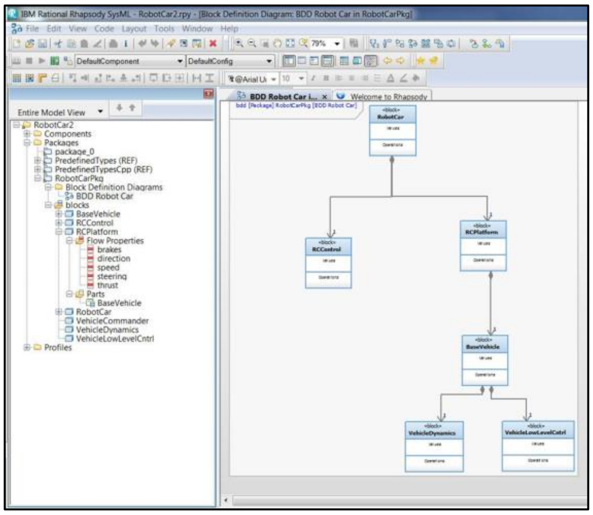


Figure 5. The mediation flow demonstrated in the SPRINT project.

To facilitate mediations, a configurable platform has been developed to serve the SPRINT and the DANSE projects based on the IBM Jazz platform [8]. SMC allows the user to create RDF ontologies, import them from various sources on the Internet, and alter them using the Protégé [28] ontology editor. Rule ontologies, also editable with Protégé, are distinguished entities of SMC: they are associated with a pair of ontologies. A web interface allows the administrator to plan mediation links among repositories holding RDF datasets of model instances. Each repository is associated with its own ontology and is expected to comply with it. Tools such as Rhapsody, and SystemModeler can export and import their model projects to and from the SMC. Once that happens, SMC triggers mediation steps, which create for each input model a corresponding output model, like the AML and UML model instances in the example above. The mediation is carried out by an implementation of a mediator. The mediator is an interpreter of the mediation rules ontology. When configuring a mediation link between two ontologies, that link must be associated with both the rules ontology and the mediator that will execute it. Different mediators have been developed on the SMC, some by IBM, some by other vendors.

A simplified diagram of a mediation "network" as demonstrated in the SPRINT project [29-33] is depicted in Figure 5, in which the mediations are drawn as arrows, labeled in yellow and red circled numbers according to a scenario in which an original SysML model in Rhapsody is exported (1) to the first RhP repository as an RDF model. The RhP repository is mediated (2) to an intermediate repository, BSO, whose ontology represents the common structural concepts of the three ontologies for the three tools in this scenario. In the following mediations, (3) and (4), this intermediate model is mediated in parallel to Desyre and to the Modelica tool SystemModeler, in which the model is expressed in the Modelica language. The RDF model is also imported (5) to the tool. The SystemModeler engineer now adds new components to the model and exports it back to the SMC, following again the mediation steps in the reverse direction (labeled in red). Finally, the modified model, including the additions and modifications made by the SystemModeler engineer, is imported into Rhapsody.


IBM Rational Rhapsody SysML - RobotCar2.rpy - [Block Definition Diagram: BDD Robot Car in RobotCarPkg]

```
Entire Model View
RobotCar2
  Components
  Packages
    package_0
    PredefinedTypes (REF)
    PredefinedTypesCpp (REF)
    RobotCarPkg
      Block Definition Diagrams
        BDD Robot Car
      blocks
        BaseVehicle
        RCControl
        RCPlatform
          Flow Properties
            brakes
            direction
            speed
            steering
            thrust
          Parts
            BaseVehicle
        RobotCar
        VehicleCommander
        VehicleDynamics
        VehicleLowLevelCntrl
  Profiles
```

Figure 6. The Rhapsody model browsing view, with the "explorer" on the left side panel showing the hierarchical structure of the model.

3. Developing the Rhapsody Ontology

The tools, intermediate ontologies, and rule ontologies that were developed in SPRINT are published as appendices in the SPRINT final report [30]. The purpose of this section is to convey the rationale behind the construction of the Rhapsody ontology. The ontology is intended not as a formal description of the modeling language implemented in that tool, but mainly to serve semantic mediation of models from that tool to models in other ontologies.

When facing this challenge, there are several forces that can pull the results in different directions:

1. Being a SysML tool, adopting a SysML OWL ontology seems like a good idea. However, there is no such specification, but being a formal standard of OMG, the SysML ontology is formally defined using eCore (an OMG MOF standard) [34]. It is possible to convert the eCore meta-model to OWL. The developer of the export/import code must figure out how to implement these services using the tool's internal data access facilities.
2. One can consider as most critical the technical difficulty of extracting a model from the tool to RDF and the ability to merge the RDF back to an existing model (or the simpler case of recreating a model in the tool from an RDF model). In this case, it may be tempting to simply create an ontology which reflects on the internal tool data structure. For instance, many elements in the Rhapsody internal data structure are variants of a basic building block, through some properties. Hence, it would be simpler to reflect that basic building block in the ontology and define the different properties which assign that element the different SysML roles in the model.
3. One can build the resulting ontology by using the domain concepts explicitly, in our case SysML. That may look similar to the first force listed above, and it may really be that, in case an ontology for SysML is indeed written as an OWL model. Yet, a SysML ontology may not be exactly what Rhapsody implements, or for that matter any other tool that may claim to be a SysML compliant tool. Hence, it would be better to first worry about our tool and the "language" it interprets, rather than whether it is the standard SysML or its own "standard".

Experience with the first approach above has shown us that the resulting ontology is rather complex and rich with information that is overwhelming. That is all depending on the details of the eCore metamodel. In SPRINT, the SystemModeler OWL ontology has been created from an eCore model, published as appendices of the final project report [30]. Yet, the eCore model here has been designed with the third criterion above in mind, and was lean to produce also a leaner OWL ontology.

The ontology needs to be lean so it can be used for mediation, not for validation of the resulting RDF models. Hence, the OWL capabilities to define *restrictions* are not required and are not expected to be used as part of the mediation interpretation of the rules. The ontology should convey the structure of the models using just some of the OWL concepts, including *objectProperty, dataProperty, domain, range,* and *sub-relations* (among classes, and among properties).

Another example is an ontology that followed the second criterion, which was easy to develop and generate RDF from. This is a "dump" or "serialization" of the internal model data structure, making it easy to extract and embed back into the tool. Yet, mediating this to other models of other tools would be difficult, and an engineer who is used to the tool's GUI and the tool's explorer view of the model (see Fig. 6) that RDF will be hard to follow and understand.

Hence, the approach we took has been to reflect all the concepts that the engineer is faced with when using the tool, reflecting the tool's "language". Looking at the explorer of Rhapsody for a certain model, the concepts include things such as "Block", "Part", "Port", "Connector", "Attribute", "Type", "Stereotype", and so on. The Rhapsody ontology defines all these concepts as OWL classes. Next, the relations among elements of these types may be of ownership to facilitate the tree structure of the model, as shown in the explorer hierarchical tree view. Yet, there are also other references among the entities. For instance, a "Part" references a "Block", which is its "type". Contrary to that, when a "Block" contains "Parts", that relation is an ownership relation. Still, both relations will look the same in the ontology, as they are represented as *owl:ObjectProperty*.

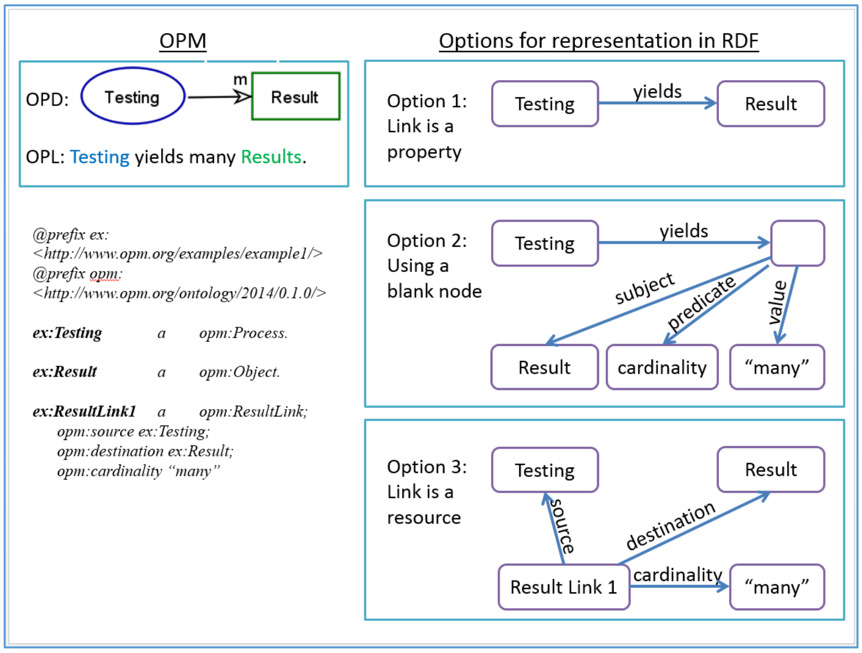
The next section describes the OPM ontology development. The Rhapsody tool can also model behavior of "Blocks", much like the "Process" aspect of the OPM models, yet the Rhapsody behavior is defined through state-charts state machines showing the reaction of the block producing output signals depending on its state and input signals. That behavior aspect will be coded as "contracts" (not describe in this paper) for the Desyre tool, and analytic equations in Modelica. All these tools can produce simulation objects which can be used in system evaluation and testing, and which are written in the standard FMI (Functional Mockup Interface) [35]. The SMC approach to that has been therefore to ignore the internal formal behavior description, be it contracts, equations, or state-charts, and associate (with a link) the RDF resource representing a Block (in the Rhapsody case) with the FMI binary object which is kind of a “blob” that FMI tools can work with.

4. Developing the OPM ontology

OPM is a holistic methodology for modeling complex systems of all kinds. Consisting of a minimal ontology of stateful objects and processes that transform them, with simple syntax and well-defined semantics, OPM enables representing the system’s function, structure, and behavior in a set of hierarchically organized diagrams of the same and only type—Object-Process Diagram (OPD). Each OPM element (thing, i.e., object or process, or link) has precise semantics, allowing validation of a model using restrictions enforced by the OPM modeling tool OPCAT [24] while building and executing the model. OPCAT provides a simulation module as well as textual representation in Object-Process Language (OPL).

For defining OPM ontology in OWL, considerations similar to those of the Rhapsody ontology construction were taken. First, the scope of the ontology was determined. Then, the data preservation level was decided. Finally, different approaches of elements representation in RDF were considered. The work aimed not only to fulfill the requirements of SMC, but also to start integration of OPM into the Semantic Web. Therefore, our objectives were established to preserving maximum knowledge while allowing efficient data retrieval. Whenever a narrower representation is needed, information may be inferred from the complete model. The completeness of the ontology has been examined by round-trip transformation of OPM model from OPCAT to RDF and back, using the ontology-based export and import modules [15, 16].

The scope of the ontology was determined to represent all the semantically meaningful information and only that information. OPM models consist of entities—things and links. Things are objects and processes, which have relations among them expressed as structural and procedural links. In the graphical representation in object-process diagrams (OPDs), some visual layouts also express semantics. For example, the relative vertical positioning of subprocesses in the context of an in-zoomed process determines their order of execution, from top to bottom. Hence, while in general visual attributes such as positional coordinates and size of elements were kept out of the ontology, meaningful visual relations, such as vertical position, were included. For example, the property “is before” describes the relation between two subprocesses, where the earlier subprocess is positioned above the later one.



```
@prefix ex:
<http://www.opm.org/examples/example1/>
@prefix opm:
<http://www.opm.org/ontology/2014/0.1.0/>

ex:Testing        a      opm:Process.

ex:Result         a      opm:Object.

ex:ResultLink1    a      opm:ResultLink;
    opm:source ex:Testing;
    opm:destination ex:Result;
    opm:cardinality “many”
```

Figure 7. Three options for representing OPM links in RDF

The entire OPM model is expressed through a hierarchical set of OPDs. The tree hierarchy is generated by process in-zooming: Each new-diagram in-zooming operation creates a child OPD. All the OPDs are self-similar, in that all OPDs, regardless of their depth level, use the compact ontology of stateful objects and processes that transform them, as expressed by the various links. The hierarchy helps cope with the human limited channel capacity [36], but in principle, even the most complex OPM model can be expressed in a single, flat diagram. Two models with different hierarchical structure may be semantically identical, since the processes in the single flat diagram can be grouped in many ways. Therefore, the hierarchical structure of the OPM model is not included in the ontology.

The level of data preservation within the scope corresponds with the OPM specifications in the OPM 19450 ISO standard – publically available specification [37]. Some elements from the specifications were not implemented in the ontology because they cannot be realized and tested in OPCAT yet. OPCAT has capabilities that are not mentioned in the OPM standard, for instance special properties of elements, which were not included in the ontology. However, new elements may be easily added to the ontology without affecting the existing definitions.
Some OPM element representations in OWL is straightforward and intuitive. For example, objects and processes are clearly resources with properties. Links, however, may have different definition options. Several methods were examined, from representing links as properties of objects and processes, through using blank nodes for compound statements, to creating a resource for each link instance. Some of the options considered are illustrated in Figure 7. Representing each link as a resource enables better preservation of data which describes the link itself, e.g., cardinality and tags. Inheritance of resource types and properties is used in the OPM ontology. *opm:Process* and *opm:Object* inherit from *opm:OPMThing*, and all the links inherit from *opm:Link*.

5. Summary

The promise of semantic mediation in model-based system engineering is that the content of models in different modeling languages, built using different modeling tools, can be shared with other tools. Our approach to achieving model interoperability is to use ontologies for providing a common semantic basis to the models. The models share a common RDF format, and ontologies are also expressed as RDF graphs using the OWL specification. We discussed the commonalities and differences between the semantic mediation approach and that of the OSLC initiative, which leads the way with the application of the W3C linked data to systems engineering. Using the semantic mediation container (SMC), models can be mediated from some representation based on one ontology to another representation based on another ontology. Only those elements that are common to the two mediated languages and the tools in which they are expressed can be mediated. An intermediary ontology for these common concepts can serve as a practical (though limited) "lingua franca". Accordingly, a tool that can export its model content in the RDF structure of SMC and associate it with ontology is capable of performing the first step of interoperability. The rest of the data flow is accomplished through semantic mediation. We discussed the considerations of building an ontology for a tool using examples of two different tools—Rhapsody for SysML and OPCAT for OPM—both standards in the common domain of conceptual modeling for system design, the foundational tenet of model-based systems engineering. For Rhapsody, the benefits gained from this step were demonstrated in the SPRINT EU project, and are also applied in the DANSE EU project. For OPCAT this is only the first step, and mediation to other ontologies is yet to be developed. As more tools take this initial step, the interoperability among the tools with respect to model sharing will bring new benefits to the product development lifecycle of complex systems.

6. Acknowledgments

This work is supported in part by the (completed) DANSE (Grant agreement no: 287716) EU 7th framework programme project, and the completed SPRINT (Grant agreement no: 257909) EU 7th framework programme projects, and the ongoing H2020 EU project PSYMBIOSYS (Grant agreement no: 636804).

References

1. A. L. Sangiovanni-Vincentelli. Quo Vadis, SLD? Reasoning about the Trends and Challenges of System Level Design. Proceedings of the IEEE. **95**:3.March 2007.
2. NASA STEP Central, ISO 10303, *http://step.nasa.gov/*.
3. EDIF Standard for Electronics Data Interchange, *http://www.princeton.edu/~achaney/tmve/wiki100k/docs/EDIF.html*.
4. Open Services for Lifecycle Collaboration (OSLC), *http://open-services.net/*.
5. R. Filding. Architectural Styles and the Design of Network-based Software Architectures. PhD Dessertation, 2000, University of California, Irvine. *http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm*.
6. W3C World Wide Web Consortium. Linked Data - W3C. *http://www.w3.org/standards/semanticweb/data*.
7. C. Bizer, T. Heath and T. Berners-Lee. Linked data - The story so far. *Int. J. Semant. Web Inf. Syst.* 2009;**5**:1-22.
8. The IBM Jazz Integration Architecture (JIA). *http://jazz.net/projects/DevelopmentItem.jsp?href=content/project/plans/jia-overview/index.htm*.
9. SPRINT - Software Platform for Integration of Engineering and Things. *http://www.sprint-iot.eu/*.
10. DANSE - Designing for Adaptability and evolutioN in System of systems Engineering. *http://www.danse-ip.eu/home*.

11. Rational® Rhapsody®. *www-142.ibm.com/software/products/us/en/ratirhapfami/*
12. SysML™ specification version 1.3. *http://www.omg.org/spec/SysML/1.3/*.
13. Wolfram SystemModeler. *www.wolfram.com/system-modeler/*.
14. Modelica Language Specification. Version 3.3. Modelica Association. 2010. May 9. *https://www.modelica.org/documents/ModelicaSpec33.pdf*.
15. Unified Profile for the Department of Defense Architecture Framework (DoDAF) and the Ministry of Defence Architecture Framework (MODAF). OMG Document Number: formal/2013-08-04. *http://www.omg.org/spec/UPDM/Current*.
16. D. Dori. Object-Process Methodology: A Holistic Systems Paradigm. Springer, Berlin, Heidelberg, New York, 2002.
17. S. Jacobs. Translating OPM System Models to RDF Format for Their Integration into the Sematic Web. M.Sc. Project Thesis. Information Management Engineering, Technion, Israel, August 2014.
18. S. Jacobs, N. Wengrowicz and D. Dori. Exporting Object-Process Methodology System Models to the Semantic Web. IEEE International Conference on Systems, Man, and Cybernetics. 2014.
19. Y. Grobshtein and D. Dori, Generating SysML Views from an OPM Model: Design and Evaluation. Systems Engineering, 14 (3), pp. 327-340, 2011.
20. Y. Yaroker, V. Perelman, and D. Dori. An OPM Conceptual Model-Based Executable Simulation Environment: Implementation and Evaluation. Systems Engineering, 16(4), pp. 381-390, 2013.
21. A. Sharon, O. de Weck, and D. Dori, Improving Project-Product Lifecycle Management with Model-Based Design Structure Matrix: A joint project management and systems engineering approach. Systems Engineering, 16 (4), pp. 413-426, 2013.
22. A. Sharon and D. Dori, A Project-Product Model-Based Approach to Planning Work Breakdown Structures of Complex System Projects. IEEE Systems Journal, 2014, Digital Object Identifier: 10.1109/JSYST.2013.2297491
23. Y. Mordecai and D. Dori, Model-Based Risk-Oriented Robust Systems Design with Object-Process Methodology. International Journal of Strategic Engineering Asset Management, 1(4), pp. 331-354, 2013.
24. N. Wengrowicz, Y. J. Dori, and D. Dori, Transactional Distance in an Undergraduate Project-based Systems Modeling Course. Knowledge-Based Systems 71, pp. 41-51, 2014.
25. J. Somekh, G. Haimovich, A. Guterman, D. Dori, and M. Choder, Conceptual Modeling of mRNA Decay Provokes New Hypotheses. PLoS ONE 9(9): e107085. doi:10.1371/journal.pone.0107085.
26. D. Dori, I. Reinhartz-Berger and A. Sturm. Developing complex systems with object-process methodology using OPCAT. Lect. Notes Comput. Sci. 2813:570-572. 2003.
27. Turtle - Terse RDF Triple Language. W3C Team Submission 28 March 2011. *http://www.w3.org/TeamSubmission/turtle/*.
28. Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine. Protégé. *http://protege.stanford.edu/*.
29. U. Shani, et al. Architectural principles for Internet of System Design. A SPRINT deliverable Feb 16, 2014. *http://www.sprint-iot.eu/public_deliverables/D_5_7.PU.Architectural%20principles%20for%20Internet%20of%20System%20Design%20and%20the%20IoT.pdf*.
30. U. Shani, Final Report. A SPRINT project deliverable. March 17, 2014. *http://www.sprint-iot.eu/public_deliverables/D_5_11.PU.Final%20Report.pdf*.
31. U. Shani, Daniel Wadler, Michael Wagner. Enginering Model Mediation Which Really Works. *The 7th National Conference INCOSE_IL*. March 2013.
32. P. Aronsson, O. Tronarp, D. Hedberg. A Collaborative Platform for Systems Engineering Tools over the Internet with Connections to Wolfram Systemmodeler. *7th MODPROD Workshop on Model-Based Product Development*, February 2013.
33. U. Shani and A. Landau, Tools Interoperability Platform for Model-Based Systems-Engineering. *MBSDPTI workshop SECOOP'13*, July 2013.
34. OMG Meta Object Facility (MOF) Core Specification. Version 2.4.1, June 1, 2013, OMG Document Number formal/2013-06-01. *http://www.omg.org/spec/MOF/2.4.1*.
35. FMI for Model Exchange and Co-Simulation. Version 2.0 specifications. July 25, 2014. *https://www.fmi-standard.org/downloads#version2*.
36. D. Dori, Words from Pictures for Dual Channel Processing: A Bimodal Graphics-Text Representation of Complex Systems. Communications of the ACM, 51(5), pp. 47-52, 2008.
37. D. Dori, The Maturation of Model-Based Systems Engineering: OPM as the ISO Conceptual Modeling Language Standard. MIT SDM Webinar, 2014. https://groups.google.com/forum/#!topic/astewg/0eGz3-fWHtQ