Can ontologies prevent MBSE models from becoming obsolete?

Uri Shani
IBM Research-Haifa
Haifa University Campus, Mount Carmel
Haifa Israel
shani@il.ibm.com

Abstract—Model-based systems engineering provides a soft, manageable, and query-able representation of product design and lifecycle. However, at the same time, it creates a discontinuous space of data that does not integrate, constantly becomes outdated, and has to be upgraded as tools evolve. Models that are not maintained become legacy, and then become obsolete and useless. We look at how ontologies that follow the semantic web technology for data representation can create interoperability among the modeling tools, support model reuse, and fight the aging and obsolescence of models.

Keywords—Ontology; MBSE; modeling; obsolescence; reusability; interoperability

I. INTRODUCTION

Ontologies represent data facts and the associated data rules. Therefore, they can provide a self-contained description of data, a closure that can withstand the test of time. Ontological rules are metadata that can be considered the semantics of the data. In model-based systems engineering (MBSE), each model is a data set, and its semantics are its metadata.

Every modeling tool uses its own modeling language dialect, and its own proprietary model and schema for data representation. This holds even when the tool claims to serve a certain standard modeling specification, for example SysML[1], or Modelica[2]. MBSE provides a soft representation of product design and can integrate into the product lifecycle tool chain. Yet the integration does not work, since tools are largely isolated from each other. Representing the models in tools and their formal language rules in a unified formal specification can enable the sharing of models among these tools.

The progress and evolution of tools, languages, and specifications can continue to best serve the systems engineer community. However, past work does not have to become useless legacy; it should be fully reusable and provide recoverable information about early product design considerations.

In this paper, we review Open Services for Lifecycle Collaboration (OSLC) [3]. OSLC takes a bold step in opening up the sealed semantics of a tool, to enable sharing and linking its content model with other design and lifecycle tools serving the product. OSLC adopted the semantic web, taking the Resource Description Framework (RDF) [4] format to represent data in models as a network of resources, and defines the semantics of these resources via the Web Ontology Language (OWL) [5]. OWL rules are used at the very basic level, which serves OSLC well. We [6][7] extended the use of OWL ontologies to represent several modeling tool languages, so that full models maintained in different tools could be represented in RDF, creating a semantically sound representation of the models. We limit ourselves to the OWL semantics because that is the only standard specification necessary as a base. All other tools can represent their languages with that specification, so that a model compounded with its ontology can be considered a closure that is fully self-describing.

We developed the Semantic Mediation Container (SMC) for our systems engineering projects SPRINT [8] and DANSE [9]. In these EU projects, we used SMC to create bridges among modeling tools to share models and establish a collaborative environment and tool chain. We further use this tool to facilitate collaboration and model sharing in product service systems (PSS) in the PSYMBIOSYS H2020 EU project [10].
Semantic mediation as applied with the SMC shows that different models designed using different modeling languages can be mediated to each other, reused [11], and thus essentially become tool-independent. A tool-independent model with its ontology as a semantic description can be meaningful also after the tool used to create it reaches the end of its life. It may be replaced with a newer major version, or abandoned for a different, standard tool and methodology.

A model will keep its value and usefulness through periods of tool evolution if it can be mediated for later tools that succeed the one used to create the model. Moreover, even without a tool, it should be possible to interpret that model and expose its semantics in terms that make sense to engineers.

We suggest that ontology is a candidate for a universal language to represent modeling data. This would allow the models to withstand time through reuse among different tools and different languages as they evolve and progress. They would basically fight their own obsolescence.

In the next sections, we briefly describe our proposed ontology language and bring examples from experience in mediating models.

II. MODELING WITH ONTOLOGY

Representing a model with ontology consists of several parts: A notational, syntactic representation, the modeling language rules, and the model instance representation with concepts defined in those rules. That can also apply to any modeling system or tool such as Rhapsody, where models are presented to the engineer visually via diagrams and a corresponding explorer to view and explore the hierarchical model structure. The model itself is represented and persisted as a Rhapsody project, in which it has an internal and proprietary structure. The specific modeling language the tool uses is SysML, but there is no formal representation of this language in the tool (compliance with that standard is maintained via the visual diagrams and the elements position in the hierarchical explorer view).

When we represent a model with ontology, we use RDF notations in one of several different but interchangeable syntaxes, such as XML, Turtle, Triples and N-Triples, JSON, and others. The language rules are coded using the OWL language, which is also written in RDF, so that the notations have the same formal presentation for both the rules and the instance models. OWL is the meta-language of all languages, and as such has its own rules, which are part of the OWL specifications that we cannot fully review within the scope of this paper.

RDF is a representation of resources in a graph consisting of arcs and nodes, where nodes are resources or data items such as numbers, strings, and dates. Arcs define relations among the nodes. In an ontology graph, nodes are resources representing classes and properties (i.e. concepts of some modeling language), and arcs define the relations among these concepts. To define these concepts, OWL uses other concepts from the OWL language. The OWL [12] ontology rules are logical statements about the relations among these concepts and are based on descriptive logic (there are actually several levels of compliance with logical capabilities of the models). On that basis, we can apply logical inference over ontology to conclude (i.e., infer) new relations.

We distinguish the rules that define the concepts of a modeling language for a specific tool, such as Rhapsody® [13], which implements the SysML standard, from the facts that comprise an instance model. These “facts” are resources defined using resources that belong to the ontology. The same ontology can be used to describe many different model instances. In OWL terminology, the set of concepts are considered the “TBox” while the facts are the “ABox”. When applying inference rules over a model, we may infer new relations among the facts using the OWL statements defined in the ontology that represents the modeling language in the TBox.

To demonstrate that, we depict in a few figures the application of ontology to model a SysML project in Rhapsody. Figure 1 shows a SysML diagram of a model with a few components and connectors between them, as the elements of a package. Figure 2 shows the hierarchical explorer view of this structure. The internal data structure representing this model is proprietary, a component of the Rhapsody commercial tool. Figure 3 is an RDF graph of this model according to an OWL ontology that we developed. The model graph has resource nodes representing elements of the model, and the arcs are relations among these elements. These nodes are classified (having the ontology
property “Class”) according to this ontology. Other nodes in the ontology are classified as “properties”. These resource nodes are also concepts, which label the arcs among model nodes. All these concepts are defined according to the OWL specifications, in yet another RDF graph. The formal description in RDF of the Rhapsody ontology we developed can be found in the Appendix of the SPRINT final report [14].

Figures 4-6 show another example for a different tool and language – Modelica [2] and the SystemModeler® [15] tool. Figure 4 illustrates a simple model diagram of two components and a connector. Figure 5 shows the hierarchical explorer view of these elements. Figure 6 is an RDF graph representing that model according to an OWL ontology developed to represent the Modelica
language dialect as implemented in that tool — for the specific version we used during the SPRINT project.

As with the Rhapsody example above, this OWL ontology is also an RDF graph. Its full details are also included in the Appendix of the SPRINT final report [13].

### III. Semantic Mediation

Let's assume that we represent the modeling language as OWL ontology in RDF graph, and models in that language in RDF graphs as well, viewed as the ABox facts of that ontology. Having that, we are no longer dependent on the originating tool of the model, nor its modeling language. We should be able now to interpret this model in a useful way, even without that tool.

In SPRINT (and then in DANSE) we developed semantic mediation to bridge the ontologies of different languages and tools. In SPRINT we were able to reuse models that originated in Modelica or SysML. We used them, changed them, and shared them in any of these languages using specialized (commercial) tools designed to implement those languages. That service provided tool interoperability, and is demonstrated in Figure 7. The figure shows semantic mediation that enables sharing a model among three tools, including Rhapsody and Modelica, using an intermediary language representation shown in the middle. Clearly, each of these tools needed an adapter, a plugin or extension to its code. The code uses an export function to convert the internal proprietary representation into RDF, and an import function to convert the RDF back to that internal representation.

That capability has been expanded in DANSE and demonstrated over what can be a representative case for tool and language evolution. SysML is in fact an extensible language through the notion of profiles. In DANSE, we dealt with the design of System of Systems [16], and UPDM (Unified Profile for DODAF and MODAF) [17] is a standard extension of SysML.

In DANSE, we extended the SysML ontology of the Rhapsody tool with UPDM concepts. UPDM models developed on Rhapsody were exported into RDF graph representation in that ontology. As shown in Figure 8 below, Semantic Mediation can mediate the UPDM models to the SysML simplified variant, creating for each UPDM project a corresponding SysML project on the tool, so that some analytics capabilities developed to work with SysML models could also be applied to the converted UPDM models [18].

The DANSE use case demonstrated that model representation in RDF with the associated ontologies provides a means to maintain them across language evolution and tool revisions.
Shani et al. [19] expanded and reported the span of languages using ontology, to provide interoperability of the OPCAT [19] systems design tool. Work with this tool stopped short of mediating, but provided an adapter to convert models represented in the OPCAT specifications to an RDF, according to an ontology describing the OPM [21] – the language of the OPCAT tool. See Figure 9.

The OPM language and its OPCAT tool are public domain, an IEEE standard, and an open source tool. The tool shares a human-readable representation of the model that reflects the diagram graphics representation of the model. The two representations are interchangeable. Yet, it is a private notation of the tool and not possible to share with other modeling languages and tools. However, with the use of ontology and RDF, it has become tool independent according to the principles discussed in this paper.

The paper [19] also discusses the considerations of developing ontologies for different tools – for both the Rhapsody SysML ontology and the OPCAT OPM ontology. Our work sheds light on the issues and decisions facing this development, to properly reflect on the semantics of the language in the ontology.

IV. SIMPLE MEDIATION EXAMPLE

An example of mediation between two simple ontologies may convey the basic idea, although its simplicity should not be misleading regarding the value of this method when applied to real and complex modeling languages and specifications, as discussed above.

Fig. 8: Mediating a UPDM (SysML extension) model to a SysML model represented here in the Rhapsody structure explorer view

Fig. 9: An Object Process Diagram (OPD) of the OPM language showing the three basic concepts of this modeling language: Object, State, and Process (from [20])

Fig. 8: Mediating a UPDM (SysML extension) model to a SysML model represented here in the Rhapsody structure explorer view

Fig. 9: An Object Process Diagram (OPD) of the OPM language showing the three basic concepts of this modeling language: Object, State, and Process (from [20])
We describe two similar ontologies representing two “languages” that we term SML and UML. SML may represent a tiny “SysML”, and UML may represent a tiny Unified Modeling Language used to model software designs. Figures 10 and 11 depict the simplified ontologies defining the concepts of these two languages.

In Figure 10 we define the concepts Type and Object as classifications (OWL classes), and the terms hasType and hasObject as properties, which connect instances of an Object with instances of a Type in complementing ways.

Similarly, the SML ontology defines the concepts of Block and of Part as OWL classifications, and the terms hasPart and hasBlock to link instances of a Block with an instances of a Part, and vice-versa. To save space, Figure 11 does not show the UML ontology, but a model instance using that ontology.

We assume that the reader can infer it from that model example.

We show the semantic mediation rules that bridge these two ontologies in Figure 12. Here, we see a simple set of ontology rules that equate concepts of these two ontologies in a very simple way. Applying these rules on the SML model of Figure 11 would result in a very similar model for the UML ontology, shown in Figure 13. The simple equivalent rules are the easiest to understand. A deeper discussion of the mediation rules is provided in [22]. That paper is published in this same IEEE Systems Conference.

SUMMARY

Semantic mediation is not about linked data and the
management of modeling resources in the Application Lifecycle Management (ALM) tool chain. It relates to model reuse, interoperability among MBSE tools, and exchanging and sharing models among them.

That is not an easy task, as these languages do not have a full overlap with each other, and reuse requires bridging the ontological metadata – which represents the “semantics” of the modeling language. The value of a new language is not in being equal to another language, but rather in its differences and improvements over other languages.

The same ideas hold for the same tool and its language as it evolves over time. There may be some meaningful functionality improvement in the tool to justify a major revision.

We claim that when looking at a single tool, its language, semantics, and model instance representations, there is no easy way to have old models remain compliant with the new versions of the tool, or new revisions of its modeling language. At some point, models will stop migrating and will be abandoned. They may become legacy and eventually lose their relevance, until their legacy turns into obsolescence.

Based on our experience working with ontologies to model modeling languages, and using the semantic web techniques and tools to represent such models, we postulate that resorting to the proposed representation may enable to keep obsolete or legacy models alive over time.

It is easier to mediate a model from old revisions of the modeling system than to bridge different languages. The nature of evolution is that some features are replaced with more sophisticated ones, and new ones are added. An old model may need to adopt the modified features, and may not need to use any of the new features – thus making most of its content reusable and valuable with the newer tools. Our example of mediating UPDM to SysML shows backwards migration of a model; however, we believe it demonstrates this point.

The model representation in RDF, with OWL ontologies also represented in RDF, has an obvious advantage over the proprietary and individual representations of modeling tools. In the latter, their semantics are informally embedded within the guts of algorithms and data flows implemented in some programming language. The compound data of the ontology and its instance models makes them a whole that can be deciphered over time and saved from obsolescence.

We consider this paper a “thought paper”, not a magic solution to a difficult problem. We hope that through the examples presented above, we showed that a common representation, which is self-explanatory and has a semantic foundation for all models, is a viable premise.

ACKNOWLEDGMENT

Supported by the PSYMBIOSYS EU H2020 project grant #636804.

REFERENCES

[1] SysML: The Systems Modeling Language (SysML) based on UML 2.0, http://www.omg.org/sy

[2] Modelica Language Specification, Version 3.3, Modelica Association, 2010, May 9, https://www.modelica.org/documents/ModelicaSpec33.pdf.

[3] Open services for lifecycle collaboration (OSLC), http://open-services.net/.

[4] RDF: Resource Description Framework, http://www.w3.org/RDF/.

[5] OWL: Web Ontology Language. "OWL 2 Web Ontology Language Document Overview". W3C. 2009-10-27. http://www.w3.org/TR/owl2-overview/

[6] U. Shani and A. Landau, Tools Interoperability Platform for Model-Based Systems-Engineering, MBSDPTI workshop SECOOP'13, July 2013.

[7] U. Shani, D. Wadler, and M. Wagner, Engineering model mediation which really works, The 7th National Conference INCOSE_IL, March 2013.

[8] SPRINT - Software Platform for Integration of Engineering and Things, http://www.sprint-iot.eu/

[9] DANSE EU FP7 project, “Designing for Adaptability and evolution In System of systems Engineering,” 2011-2015, www.danse-ip.eu

[10] PSYMBIOSYS EU H2020 project, “Product-Service SYMBIOtic SYStems,” 2015, http://www.psymbiosys.eu/

[11] Uri Shani and Henry Broodney, “Reuse in Model-Based Systems Engineering,”

http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=7112393 3-16 April 2015, Page(s) 77 – 83, Vancouver, BC, DOI: 10.1109/SYSCON.2015.7116732

[12] Dean Allemang and Jim Hendler, “Semantic Web for the Working Ontologist – Effective Modeling in RDFS and OWL," Elsevier, 2011, ISBN 978-0123735560, 2011

[13] Rational® Rhapsody®, www- 142.ibm.com/software/products/us/en/rathpsyfami/

[14] 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.

[15] Wolfram SystemModeler, www.wolfram.com/system-modeler/

[16] Maier, M. W. “Architecting Principles for Systems-of-Systems.” Proceedings of the 1996 Symposium of the International Council on Systems Engineering.

[17] UPDM: "Unified Profile for DoDAF and MoDAF," http://www.omg.org/spec/UPDM/

[18] Uri Shani et. Al., “DANSE Prototype - Final Report, DR 8.3,"http://www.danse-ip.eu/home/pdf/danse_d8.8prototype_final_report.pdf

[19] Uri Shani, Shmuela Jacobs, Niva Wengrowicz and Dov Dori
“Engaging ontologies to break MBSE tools boundaries through semantic mediation,” 2016 Conference on Systems Engineering Research

[20] Dori, D., Linchevski, C., Manor, R., & Opm, O. M. (2010). OPCAT—An Object-Process CASE Tool for OPM-Based Conceptual Modelling. In 1st International Conference on Modelling and Management of Engineering Processes (pp. 1-30).

[21] ISO 19450, 2015 ISO/PAS 19450:2015(en) Automation systems and integration — Object-Process Methodology https://www.iso.org/obp/ui/#iso:std:iso:pas:19450:ed-1:v1:en

[22] U. Shani, Marco Franke, Karl A.Hribernik, and Klaus-Dieter Thoben “Ontology Mediation to Rule them all: Managing the plurality in Product Service Systems,” SysCon’17, Montreal, April 24-27, 2017.