Exporting Object-Process Methodology System Models to the Semantic Web

Shmuela Jacobs
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Haifa, Israel
shmuelaj@technion.ac.il

Niva Wengrowicz, Dov Dori
Engineering Systems Division
Massachusetts Institute of Technology
Cambridge MA, USA;
Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Haifa, Israel
nivawen@technion.ac.il, dori@mit.edu

***Abstract*—Model-Based Systems Engineering (MBSE) is gradually becoming an acceptable good practice, especially for large-scale complex systems. The variety of modeling languages enables detailed representation of domain-specific knowledge of different aspects of a system. Semantic Web concepts provide opportunities for easier and more efficient collaboration between design and verification teams using different modeling tools. IBM Semantic Mediation Container (SMC) is a tool interoperability platform, allowing mediation and synchronization among models in different languages based on their ontologies. We propose integrating into SMC the ability to translate Object-Process Methodology (OPM), the emerging ISO 19450 Standard, which provides a holistic view of the system's function, structure, and behavior. In this paper we describe the first integration stages – defining the OPM ontology in a way that conforms to Semantic Web specifications and developing a module for exporting OPM models from their visual representation to and back from Resource Description Framework (RDF). We discuss the scope of the ontology and different approaches of element representation. We portray the implementation of the export module as an extension to OPM CASE tool (OPCAT). The ontology and export module are validated by round-trip transformation of OPM models from their visual representation to RDF and back by experimenting with OPM experts.**

Keywords—Conceptual modeling; Ontology design; RDF; Semantic Web; Semantic mediation; Object-Process Methodology

I. 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 to be considered throughout the lifecycle of a system. Traditionally, system requirements have been stated in text. However, as advantages of using visual models were discovered, model-based system engineering (MBSE) has been identified as a preferred practice for system engineering, increasing conclusiveness, traceability, and overall efficiency. Different modeling diagram types, languages, and methodologies for systems architecting and design are constantly conceived and improved to supplement or replace textual specifications. The value of these MBSE solutions increases as they serve as a basis for testing and modifying the system down the road of the system lifecycle.

The variety of modeling languages is a response to the need for specialized representations of system components and views in various domains, such as electrical, mechanical, and software engineering. A variety of software packages is available to support the different models. Several teams typically design diverse aspects of one system, each using a different modeling language and concepts. In order to maintain consistency across the various models of the same system and enable effective integration, a common platform is required. This need has been recently recognized, and platforms are being developed to cater to this need.

The Semantic Web is a relatively new approach for sharing data across the internet. It presents specifications and technologies which may be used for easier and more efficient collaboration between teams using different software tools. One of the efforts in this direction is the Open Services for Lifecycle Collaboration (OSLC) initiative [1], which is based on the World Wide Web Consortium (W3C) Linked Data [2], [3]. OSLC workgroups specify a common vocabulary that provides for interoperability between tools within and across different domains. From a wider perspective, the OSLC initiative allows for the representation not only of data, but also of actual objects, contributing to the development and expansion of the Internet of Things. 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 (REST) protocols.

Object Process Methodology, OPM [4], is a holistic methodology for modeling complex systems of all kinds. Consisting of a minimal set of building blocks 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). Modeling with OPM is especially suitable for the early stages of system architecture and design, where ideas can be captured in an agile mode by using OPM as the underlying modeling language. Each OPM

element (thing, i.e., object or process, or link) has precise semantics, allowing validation of a model using restrictions enforced by an OPM modeling tool while building and executing the model. At the detailed design stage, an OPM model may lack the ability to explicitly represent specific information needed for constructing the system. These details may be extracted from a model of the same system expressed in another language, such as SysML.

This paper presents the initial stages taken to integrate OPM models with OSLC and the Internet of Things [5], which will allow identification of the elements represented in the model by other tools. This ability is valuable for several reasons. First, a direct association between components in the OPM model and other models which elaborate them shall be available. A single click on an element in the OPM model shall reveal related up-to-date information. Second, models and data of elaborate systems in various representations can be easily transformed and combined into a single OPM model, providing a holistic view of the system's function and structure, as well as means for validating its behavior. Third, systems modeled in OPM can be tested by tools which are not directly associated with OPM. Fourth, in relation to the Internet of Things paradigm [5], management of an actual process taking place in real time can be made easier using an OPM model representing it by linking physical objects to their representation in the model. The model, presented by a suitable tool, may become a graphical user interface for monitoring and controlling the operations in the system and the status of the actual objects related to it. This research draws on two computer science and engineering fields: Semantic Web, which concerns meaningful Web-based data representation and communication standards, and model-based systems engineering (MBSE), the domain which advocates conceptual modeling for systems engineering and usage of the resulting conceptual model as the underlying blueprint for systems engineering.

II. OPM – Object-Process Methodology

Object Process Methodology [4] provides a framework based on a bimodal graphical and textual modeling language to specify complex dynamic systems. OPM defines a system using two types of elements: *things* and *links*. OPM things are *objects* and *processes*. Objects can be *stateful*, i.e., have states. A state of an object is a situation at which the objects can be. OPM links are of two kinds: *structural* and *procedural*. A structural link expresses a long-lasting relation between two things of the same persistence, i.e., between two objects or between two processes (with few exceptions). A procedural link connects a process to an object or to a state within an object (with few exceptions), indicating the nature of participation of the object in the process as an enabler, which is required but not transformed, or a transformee, which is created, consumed, or affected by that process. The time sequence of process execution is based on the relative vertical position of subprocesses inside an in-zoomed process. Each graphically expressed model fact—a link between two things—has an injective representation in a subset of English, called OPL—Object-Process Language.

Another important property of OPM is complexity management, which is achieved via its built-in abstraction/refinement mechanisms of things: folding/unfolding and out-zooming/in-zooming. These mechanisms enable the designer to abstract the system into large building blocks. These can be iteratively refined and more finely specified until a sufficient level of detail is achieved, enabling unambiguous implementation of the modeled system.

One of the most noticeable advantages of OPM, compared with other widely used languages such as SysML, is its ability to model a whole system with a single diagram type, using a minimal ontology of stateful object and linked processes as its set of building blocks. Due to this compact symbol set, the learning curve of OPM is steep. Moreover, all the OPDs in the OPM model, regardless of their level in the OPD hierarchy, are comprised of this minimal set of elements, and are therefore self-similar.

OPCAT [6] is a CASE tool for modeling systems with OPM and for simulating and testing these models. While the modeler uses the OPM building blocks to create a model, OPCAT enforces the graphical language syntax and provides an OPL textual interpretation to each meaningful editing operation of the graphic model. These means enable real-time checking of the model as it is being constructed. OPCAT implements the abstraction/refinement mechanisms and features a simulation module, which animates the system's execution, reflecting the temporal process order and the resulting object transformations, according to the process execution timeline, the existence of objects, and their presence in a certain state. The model is typically saved in an XML format.

III. Semantic Web

The Internet has provided a ubiquitous means of exchanging data and information on a global scale. The first stage in the proliferation of the Internet was publishing documents and linking them with hyperlinks serving as anchors that point and lead to other documents. The Semantic Web is a relatively recent new movement that aspires to populate the Web with machine-readable data, tagged with semantics, which will enable more meaningful information retrieval, data mining, and inference. The basic idea behind the Sematic Web is to create linked data by naming things using HTTP URI and link between them via properties. The main standards for data and properties representation are Resource Description Framework (RDF) and Web Ontology Language (OWL).

A. Resource Description Framework and Web Ontology Language

RDF [7] is a family of W3C specifications for conceptual description of information. The basic elements are *resources*, which are included in statements in the form of subject-predicate-object expressions, also known as *triples*. *Predicates* are properties of the subject resource, whose values (objects) may be another resource or a literal, such as string or integer. For example, the book Alice's Adventures in Wonderland and the author Lewis Carroll are resources. They are linked using

the predicate **author**, so the statement **Alice's Adventures in Wonderland has author Lewis Carroll** is formed.

Ontologies can be built upon RDF as part of the Semantic Web [8]: resource types and predefined properties are the resources, and the predicates describe general relations between them, including inheritance, domain, and range.

Ontologies are often represented in Web Ontology Language (OWL) [9], a RDF-based semantics formalization of description logic. In OWL, *classes* are types of resources, and *properties* are types of predicates that can link resources to each other. OWL predicates describe relations between classes and properties, which are used to infer statements from given RDF documents. However, OWL does not define constraints on statements, so it cannot be used to assess ontological correctness or completeness of a model. Moreover, it might wrongfully infer statements in order to "make sense" in the model. An example of such a case is given in [10], which presents a solution for defining constraints in the form of OSLC Resource Shapes.

In this research, RDF-OWL ontology of OPM was built using Protégé [11], an open source ontology editor and knowledge base framework. Protégé interacts with web servers that host ontologies using HTTP protocols, allowing for loading exiting ontologies and synchronizing alterations.

The export module was implemented in Java programming language. It uses Apache Jena, "a free and open source Java framework for building Semantic Web and Linked Data applications." [12]

B. *Open Services for Lifecycle Collaboration and Semantic Mediation Container*

OSLC specifications are based on the architecture of the Semantic Web. OSLC standardizes integration between different tools by establishing terminology and rules for defining resources and specifying protocols for data exchanging. OSLC advocates the W3C linked-data principle by having tools expose their internal representation of models as resources over the Internet, using the RESTful protocols. This linked-data principle creates a common interface between tools and eliminates the need to develop specific integration modules and APIs for translations between every couple of tools.

IBM Semantic Mediation Container (SMC) platform [13] is a plugin of the IBM Jazz open platform for managing the lifecycle of engineering projects. These Internet-based platforms are compliant with the OSLC specifications. SMC has been developed within two EU FP7 projects: SPRINT [14] and DANSE [15]. SMC, as well as the entire Jazz ecosystem, endorses OSLC by applying its interoperability technologies to realize a platform for exchanging data among different tools that are not based on OSLC concepts. In this platform, a system can be designed simultaneously and collaboratively by several teams in different companies and geographical locations, using multiple design methodologies and various tools. From the first stages of design through its completion to its maintenance, the integrated tools "present a single unified picture of the system being developed" [14]. The system elements are identified by unique URIs in each model. Binding two models is called *semantic mediation*: a mediator uses the ontologies and sets of rules of the modeling languages to identify elements that are semantically identical. SMC mediation is applied to RDF models, whose "language" is defined in OWL ontologies. The mediation rules in SMC are also coded in ontologies using the OWL specification. The platform's repository holds models in RDF format, as well as ontologies of the supported modeling languages and sets of mediation rules. Communication between the tools and the platform, used mainly for importing and exporting models, is done using RESTful protocols.

The process of transforming a model from one language (A) to another (B) within SMC is illustrated as an OPM model in Fig. 1. The first process, **RDF Format Exporting**, requires that the **Model in Language A** be loaded in the **Language A Modeling Environment**. Using the **Language A Ontology**, **RDF Format Exporting** yields the **Model in Language A RDF File**. The second process, **Mediating**, requires the output from the first process – the **Model in Language A RDF File**, the **Language A Ontology** and the **Language B Ontology**, and the **Language A to Language B Mediation Rules** in order to produce the **Model in Language B RDF File**, which is imported into the **Language B Modeling Environment**, using the **Language B Ontology**. The whole process is conducted over the **SMC** platform.

This paper presents the first steps taken for integration of OPM into OSLC by introducing OPM into SMC. The first step in this process is representing the OPM ontology and models in an RDF format that conforms to OSLC specifications. The second step is developing an export module of OPM models into RDF format file, as a plugin to OPM CASE tool. Further research steps include developing an import module which constructs a visual OPM model out of RDF model, and defining rules for mediation between OPM and one or more modeling languages supported in SMC.

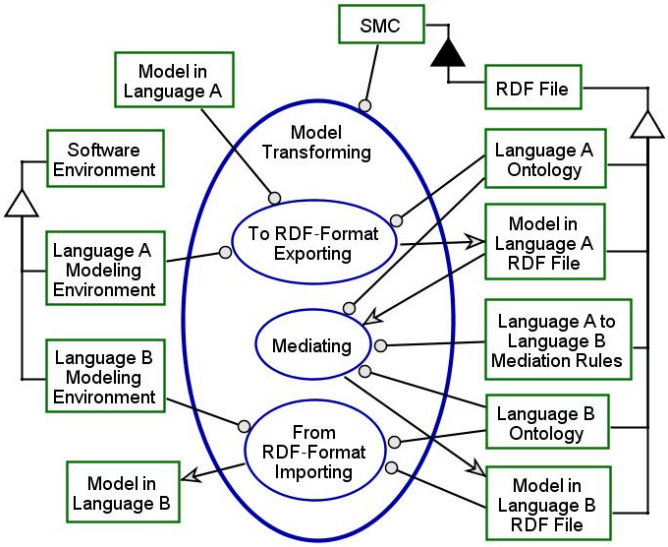

Fig. 1. An OPM model of the Model Transforming process.

IV. METHOD

Defining OPM ontology using RDF and OWL vocabulary involves depicting the structure, components, and attributes of OPM models. The ontology is used in all stages of model transforming. The export module obtains model data using OPCAT API functions and creates an RDF model based on the ontology. In OPCAT, models are represented graphically and stored in XML files as nested data structures. Transforming them into RDF format requires "flattening" them to sets of binary relations. There is not always one right way to define the ontology. Whenever alternatives for representations are valid, they should be carefully examined and tested in the mediation environment. In this paper, we examine various applicable approaches for representing compound elements, which cannot be fully expressed by one RDF statement alone. We analyze and compare theoretical aspects of the suggested designs and select solutions that fit our objectives: preserving maximum knowledge while allowing efficient data retrieval.

V. RESULTS

This section presents analysis of OPM components and discussion of various RDF patterns that can be used to represent OPM constructs. The OPM metamodel and the OPL sentences were drawn from the emerging ISO 19450 standard, the Publically Available Specification (PAS) of which is expected to be ratified during 2014.

Examples of RDF statements for suggested ontology design methods are presented in Terse RDF Triple Language (Turtle) format. The prefixes `rdf:`, `rdfs:`, and `owl:` denote the namespaces of RDF, RDFS (RDF schema), and OWL, respectively. `opm:` denotes OPM ontological terms, and `ex:` denotes resources of the example OPM model.

A. OPM Ontology Scope

Ontology is a formal specification of the concepts existing in a domain. The concepts consist of types of entities, properties that may apply to these entities, and relations among them. Three aspects should be considered for defining the OPM ontology. First, the model consists of elements, which are things—objects and processes—and links that connect them. Second, the elements have a graphical representation, which includes their appearance and layout in OPDs. Third, the diagrams have a hierarchical order imposed by in-zooming and unfolding, which are the two main kinds of OPM refinement mechanisms. These aspects are not distinctive to OPM and may be relevant to other graphical models.

OPL, the textual modality of OPM, specifies all the mandatory elements in the model and only them. This is an excellent source for validating the knowledge to be included in the ontology. For example, there is no OPL sentence for the color of the rectangle representing an object, which is merely an aesthetic attribute, but there is notation for the order of processes in an in-zoomed context, which is defined graphically by their vertical top-to-bottom positioning and textually by the reserved OPL phrase "in this order" following the list of ordered processes. Therefore, every model fact that is expressed in some OPL sentence shall be included in the ontology.

The OPM ontology for mediation should not include graphic-related data which does not reflect the semantics of the model. Most graphic attributes are irrelevant for mediation, because models in different languages have different diagram types and different visual representations of their elements. Even when considering transformation of a model between tools of the same modeling language without mediation, each modeling tool may have its own implementation of graphical appearance, for instance the direction of the Y-axis. Furthermore, when using the same tool on different workstations, screen resolution may alter the diagram display, so repositioning may be required.

The hierarchical order of the OPDs holds essential information about the behavioral aspect of the system. An OPD created by zooming into a process reveals its subprocesses. However, an OPD which presents an unfolding of an object is valuable for its content (the objects and the relations between them) but not for its place in the hierarchy.

B. OPM Ontology Classes

OPM comprises two main types of entities: things and links. Things, which are processes and objects, are the skeleton of the model. Links are the mortar that binds thing together, as a links cannot stand alone without a thing at each end. Objects and processes have names, as well as other properties, which identify them as resources of classes `opm:Object` and `opm:Process` respectively. Objects can have states, which are also resources, but are distinguished from objects and processes since they cannot exist on their own. Like objects and processes, states may serve as a source or a destination of a link.

OPM ontology includes abstract classes and inheritance relations. Although it may be sufficient to define only the concrete classes in the ontology, having abstract classes[1] and forming an inheritance hierarchy may enable useful inference. For instance, in order to retrieve all the elements that may be used as a source or a destination of a link without using inheritance, three different iterations must be performed, one for object, one for process, and one for state. By defining a superclass for objects, processes, and states, the resources of these types are specializations of the superclass so that one iteration will suffice.

Different methods for defining links in the ontology were examined. Links could have been represented as relations between two entities, but this would cause information loss because some properties relate to the link itself, such as cardinality and user-defined tags. Another approach is to describe the links as individuals, and connect them to entities using object properties: source and destination. Other properties, such as cardinality and tags, can be added to the link individual.

OPM models have general attributes, such as model name, creator and creation date. Since attributes cannot stand alone in an RDF model, an individual of the `opm:Model` class is used as their subject.

1 OWL has no notation for and does not support abstract classes. It is the responsibility of the involved tools to perform type-checking.

The basic classes in the OPM ontology are presented in Fig. 2. Bold-contoured objects denote further elaboration of subclasses (unfolding) in another OPD, not presented here. Yellow objects denote concrete classes. **Process** and **Object** are subclasses of **OPM Thing**. The class cannot be named simply **Thing** since the class `owl:Thing` is a built-in entity, which represents the set of all individuals. In order to distinguish it from things in OPM, the notation **OPM Thing** is used for a class in the OPM context. As noted, states are not things. Instead, State and OPM Thing are both subclasses of **Element**. Other classes presented in the diagram are discussed below.

C. OPM Ontology Properties

Properties are used to describe resources and the relations between them. Two kinds of properties are used in RDF models: object properties (not to be confused with OPM objects) and data properties. While the subject of both types is a resource, the value of object properties is another resource whereas the value of data property is a literal, such as string or number.

All entities in an OPM model have the data properties name and ID. Things have data properties which describe their essence (physical or informatical) and affiliation (systemic or environmental). Since the default is informatical and systemic, the data properties are `opm:isPhysical` and `opm:isEnvoronmental`, and the value is of Boolean type. If a thing is not described by one or both properties, the default is assumed. Similarly, states have the Boolean properties `opm:isInitial` and `opm:isFinal`. Other data properties describe processes' and states' minimal and maximal activation time, links' source and destination cardinality, and tagged links' forward and backward relation meaning. General model properties are attached to an `opm:Model` individual.

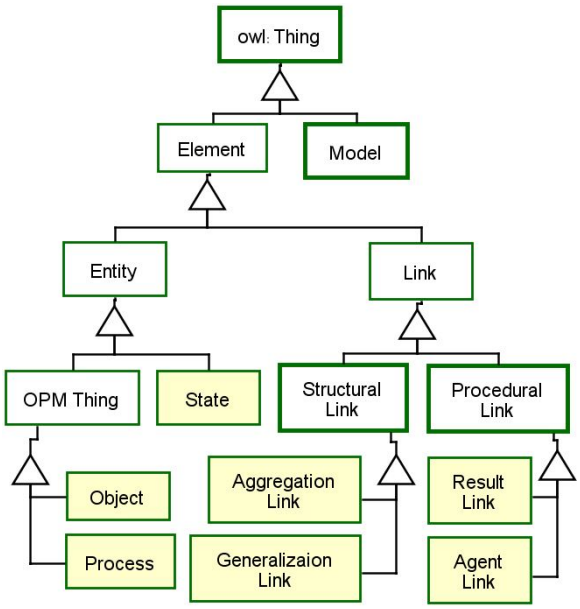

Fig. 2. An OPM model of the OPM ontology classes and their inheritance hierarchy.

Object properties describe relations between OPM entities. An object is linked to its states by the `opm:hasState` property. A process is linked to the things it contains by `opm:contains`. The order of subprocesses within an in-zoomed process context is described by the properties `opm:isBefore` and `opm:isParallelTo`. Links have the properties `opm:source` and `opm:destination` which point at the entities at the link's ends. OR and XOR relations between links are described by properties having links both as subjects and values.

D. Export Module

The export module was implemented as an extension to OPCAT The model data is obtained through OPCAT API functions. The export module begins with presenting a Save File dialog. After the user states the filename and location, the OPM ontology is loaded. Then, an RDF model is created, populated with RDF-individuals and predicates, and saved in the chosen location. The method for adding elements in the OPM model to the RDF model is hereby succinctly portrayed.

First, an RDF-individual of type `opm:Model` is created and the properties of the model as a whole are set. Next, iteration on the elements held in the model is performed. An RDF-individual is retrieved for each element, with URI which consists of the element's ID. This allows retrieval of the individual from the RDF model during the exporting process for using it as a value of properties. The type property of the element, which identifies its class, is attached to it only when the element's most specific type is identified.

First, the most general type of the element is identified, and narrowed down on each step. Properties are added to the element's representing RDF individual according to the identified class. If the element is an entity, its name is attached as an attribute of the individual. If it is a thing, the module checks its essence and affiliation. If it is an object, it is linked to its states, and the states' attributes are added. If it is a process, it is linked to its subprocesses. If the element is a link, its source and destination are identified along with cardinality, tag name, and OR and/or XOR relations with other links.

An RDF-individual may already exist for the current element in the iteration. This occurs when another element, which had already been inserted into the RDF model, has a property with the current element as the value. The Jena method `createIndividual`, which receives a URI as a parameter, creates an individual only if the URI does not appear in the model. It returns the individual represented by the URI whether it was created or already existed.

VI. Validation

The correctness and completeness of the constructed OPM ontology in RDF-OWL and the export module were verified by completing a round-trip transformation by implementing the import module in OPCAT. During the modules implementation, several simple OPM models were constructed in OPCAT. They were then exported to RDF-format files, which, in turn, were imported back to OPCAT. Some tests

revealed lack of sufficient knowledge in the RDF files, which required further development and reconsideration of ontology elements.

When the results were satisfactory, two large models were used for final evaluation. Each of these models combines all types of OPM elements. The original models were compared to the final results using two methods: comparison of OPL statements and evaluation by OPM experts. The two methods were performed considering only semantic notations: existence of things, links between them, positioning of subprocesses in an in-zoomed context, and the OPD set hierarchy.

At this stage of the research, OPM models complete the round-trip transformation satisfactorily. Further validation will be applied when integration with SMC is complete, using models mediated from other languages.

VII. Conclusion

Introducing OPM into the Semantic Web opens a wide window of possibilities, since OPM is a relatively simple and intuitive methodology which may be applied to systems of all kinds, natural or human-crafted, in all areas.

First essential steps towards integrating OPM models into the world of Semantic Web through SMC have been completed. OPM ontology definition, as well as implemented modules and supporting tools, shall be further developed to fully comply with the OPM standard. At this stage, the OPM ontology already represents the essence of OPM and allows immediate integration into SMC.

Analysis of various representation methods has produced guidelines for implementing the export and import modules and exposed issues such as the need for inference rules for querying purposes that must be addressed in the mediation stage.

The process of defining the OPM ontology in RDF-OWL, including the examination of a variety of knowledge representation approaches, is relevant not only to OPM but to all modeling languages. Research of best practices for ontology development is constantly and widely conducted as opportunities for representation of data and things on the World Wide Web emerge. The methods applied in this research can serve for construction of ontologies of other conceptual modeling languages.

VIII. Future Work

Next stages of integrating OPM into SMC shall be applied. Mediation rules between OPM and other selected modeling languages shall be defined. First, mediation shall be available between OPM and Basic Structure Ontology (BSO), which is the basic connecting node to all other languages available in SMC. Next, mediation to other, more expressive modeling languages shall be examined, in order to preserve more common data. Once mediation is available, organizations using SMC may easily include OPM models in their work, and research comparing different modeling methodologies may be conducted.

Direct communication between OPCAT and SMC shall be established, to allow easy synchronization of the model between different teams. In order to easily share and update an OPM model, without the need to reposition the elements with every model loading, data of the graphical representation and hierarchy must be preserved in the RDF file. This data may be defined in a separate ontology file, which will not be used during mediation. Furthermore, consistency of element IDs shall be enforced, so that a change in a model will not affect the identification of its elements.

During the research, some hypotheses regarding MBSE and model perception were made. These hypotheses are excellent opportunities for future research which may be conducted along with further development of OPCAT.

Acknowledgment

This research was funded by the EU FP7 VISIONAIR Project #262044 - VISIONAIR - Vision Advanced Infrastructure for Research, and Gordon Center for Systems Engineering at the Technion, Israel Institute of Technology. The authors thank Henry Broodney, Uri Shani and Ariel Landau of IBM Haifa Research Lab. This work was conducted using Protégé, supported by grant GM10331601 from the National Institute of General Medical Sciences of the United States National Institutes of Health.

References

[1] *Open Services for Lifecycle Collaboration (OSLC)* http://open-services.net/

[2] W3C World Wide Web Consortium, “Linked Data - W3C,” http://www.w3.org/standards/semanticweb/data

[3] C. Bizer, T. Heath and T. Berners-Lee, "Linked data - The story so far," Int. J. Semant. Web Inf. Syst., vol. 5, pp. 1-22, 2009.

[4] D. Dori, Object-Process Methodology : A Holistic Systems Paradigm / Dov Dori ; Foreword by Edward F. Crawley. Springer, 2002.

[5] L. Atzori, A. Iera and G. Morabito, "The Internet of Things: A survey," Comput. Networks, vol. 54, pp. 2787-2805, 2010.

[6] D. Dori, I. Reinhartz-Berger and A. Sturm, "Developing complex systems with object-process methodology using OPCAT," Lect. Notes Comput. Sci., vol. 2813, pp. 570-572, 2003.

[7] W3C World Wide Web Consortium, “RDF - Semantic Web Standards,” http://www.w3.org/RDF/

[8] D. Allemang and J. Hendler, "Semantic Web for the Working Ontologist," Semantic Web for the Working Ontologist, 2011.

[9] W3C World Wide Web Consortium, “OWL Web Ontology Language Current Status - W3C,” http://www.w3.org/standards/techs/owl

[10] A. G. Ryman, A. J. Le Hors and S. Speicher, "OSLC resource shape - A language for defining constraints on linked data," in Linked Data on the Web (LDOW2013), Rio de Janeiro, Brazil, 2013.

[11] Stanford Center for Biomedical Informatics Research at the Stanford University School of Medicine, *Protégé*, http://protege.stanford.edu/

[12] The Apache Software Foundation, "Apache Jena – Home," http://jena.apache.org

[13] U. Shani, "Conceptual and architecture principles of SoS design and semantic interoperability of systems platform and SoS design tool- net," IBM, Tech. Rep. 2014, May 6, 2013.

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

[15] *DANSE - Designing for Adaptability and evolutioN in System of systems Engineering*, http://www.danse-ip.eu/home/