Generating SysML Views from an OPM Model: Design and Evaluation

Yariv Grobshtein[1] and Dov Dori[1,2]
[1]Faculty of Industrial Engineering and Management
Technion – Israel Institute of Technology
Technion City, Haifa 32000, Israel
[2]Engineering Systems Division
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
yarivg@technion.ac.il dori@mit.edu

Abstract. Conceptual modeling is key to Model-Based Systems Engineering (MBSE) approaches. *OPM – Object-Process Methodology* and *SysML – OMG Systems Modeling Language* are two state-of-the-art conceptual modeling languages. While both languages aim at the same purpose of providing a means for general-purpose systems engineering, these languages take different approaches in realizing this goal. As each of the languages has its relative strengths and weaknesses, ways to create synergies between them are considered in this work. We propose combining advantages of each language through automatic generation of SysML views from an OPM model. To this end, we developed a new algorithm and software application for implementing the OPM-to-SysML views generation, and evaluated the results through an experiment conducted for this purpose. Our approach can benefit various stakeholders by promoting better system understanding, standardization, and improved interoperability.

Introduction

Overcoming the challenges inherent in developing ever larger and complex systems calls for transitioning from a traditional, document-centric to a modern, model-based approach. Construction and usage of a comprehensive conceptual system model throughout the system lifecycle is a key factor in successful management of contemporary systems development complexity. In this context, an important factor is the modeling language to be used for specifying the system's conceptual model.

In this paper we focus on two major state-of-the-art systems modeling languages: *Object-Process Methodology*, *OPM* (Dori 2002) and *OMG Systems Modeling Language*, *SysML* (OMG 2008, 2009). Although both languages are intended for conceptual modeling of general systems, each takes a different approach to fulfilling this mission. Previous studies (Grobshtein and Dori 2008; Grobshtein et al. 2007) investigated the characteristics of these languages in the context of general systems engineering through a comparative evaluation. In what follows, we briefly summarize these studies, which have led us to the work described in this paper.

The findings from previous works suggest that on one hand, OPM is usually more advantageous than SysML in presenting the overall picture of the system and the system's different hierarchy levels. This ability is of great importance especially in the early stages of the conceptual design. On the other hand, SysML, which is rich and comprehensive, is more suitable for modeling detailed views of some aspects, a need that usually arises during later

stages of the design process. The conclusions of these studies suggest that the selection of the modeling language should consider several factors, such as the characteristics of the specific system, the development stage, and the involved stakeholders. Obviously, subjective preferences are also an important factor.

Each of the languages has its benefits and drawbacks. For example, OPM allows holistic concurrent modeling of structure and behavior using a single diagram type, together with its bimodal graphic and text representations, namely the set of Object-Process Diagrams (OPDs) and their corresponding Object-Process Language (OPL) text of a subset of English. However, OPM sometimes does not provide for a detailed enough model as it lacks built-in dedicated support of some aspects. SysML exhibits rich and comprehensive language constructs using standard and common notation that enjoys a wide support of tools, making it widespread and common in industry. On the down side, SysML is big and complicated, so learning and using the language for both construction and comprehension involves considerable time and effort. As a result, many modelers elect to use only a relatively small subset of SysML views. The subset varies from one enterprise to another, creating a multitude of dialects that do not necessarily "talk" to each other.

Recognizing that each of the languages under study exhibits various advantages (and drawbacks) calls for finding ways to create synergies between them. While taking into account the findings from the previous aforementioned studies, we explored and developed one such way, which can potentially add value to various stakeholders. The way we present in this paper is automatic creation of a SysML model and set of views (diagrams) from an OPM model. This approach enables initial top-down OPM-based conceptual modeling, which can be translated at will to any subset of SysML views for purposes of standardization, communication, and further elaboration.

The structure of the rest of this paper is as follows: In the next section we describe the essence of the OPM-to-SysML translation and SysML view set creation, including motivation, algorithm design approach and the framework of the implemented application, along with an example. The subsequent section presents the evaluation performed for the application: We describe the experiment population and design, the results and their interpretation, discuss experiment limitations and propose conclusions. Finally, in the last section we summarize and discuss future research directions.

OPM-to-SysML Views Creation

In this section we concisely describe the OPM-to-SysML design concept and application. The section is divided into four subsections. We start in the first subsection with motivation and description of the expected benefits of the application of our approach. In the next subsection we present our design approach for creating the views. The third subsection describes the framework within which we implemented the algorithm to create a functional software application. Finally, in the fourth subsection we present a detailed specification of the algorithm and demonstrate a case in point for one type of view, the Use Case Diagram.

Motivation and Benefits

As noted, we have developed an algorithm and application for automatically creating SysML views (with their underlying model) from an OPM system model. Developing such a mechanism is interesting since it has potential benefits, which are described and argued for next.

Firstly, an OPM-to-SysML application would allow OPM users to share and present their models to other stakeholders who are familiar with the SysML notation in a relatively easy and quick manner. This advantage should not be underestimated, as common understanding of the

system and improved communication among the various stakeholders are two of the most important benefits of Model-Based Systems Engineering (MBSE) in general and of conceptual systems modeling in particular. In practice, when taking into account the wide familiarity of software engineers/systems engineers with the UML/SysML notation, it is highly likely that this model translation ability addresses a real industry need. For example, it is reasonable to assume that for some reason, a stakeholder (from inside or outside the organization) may prefer, or even requires, to get documentation of the system model in SysML. One possible scenario is when an external customer demands to get the system model in SysML, while the developing organization has used OPM.

Another potential benefit is improved system analysis and understanding. As indicated in previous studies (Grobshtein and Dori 2008; Reinhartz-Berger and Dori 2005; Peleg and Dori 2000), having dedicated aspect views may be helpful in some cases. The additional automatically generated SysML views might enable further "slicing and dicing" of the system model, so even skilled OPM users can find it useful. The additional views can be helpful for both model construction and model comprehension. This is somewhat analogous to the ability to examine and analyze a large body of data in a data warehouse using data mining techniques.

Generation of SysML views along with the underlying SysML model can also promote interoperability. A possible scenario is, for example, integration of one subsystem modeled in OPM with another subsystem modeled in SysML. This can happen due to various reasons, for example when the subsystems are developed by different organizations or when the organization is shifting from one language to the other.

Another usage scenario could be a case in which the user is interested in using some unique feature which is available only in a SysML supporting tool. In order to allow import of the auto-generated model into a SysML supporting tool, it is required that the SysML model be created in a standard format that is widely supported by tools. Fortunately, this objective is currently achievable with XMI—the XML Metadata Interchange format (OMG 2007), which is further described in the context of our work later on, in the application framework subsection.

The above benefits and possibly others indeed suggest that developing an OPM-to-SysML translation capability is potentially beneficial.

Design Approach

The main and most challenging part of constructing the OPM-to-SysML translation application is obviously the development of the translation algorithm and the engine that implements it. Taking an existing system model in OPM as input, the role of this engine is to produce a corresponding SysML underlying model with its set of diagrams. The key is to define as accurate a mapping as possible from OPM concepts to SysML ones in order to successfully match OPM elements to SysML elements in the various diagram types.

An important observation to note is that the OPM-to-SysML mapping is one-to-many in the sense that a single OPM element (entity or link) in the only OPM diagram type, OPD, usually translates to several SysML elements that belong in different SysML diagram types. Indeed, considering the OPM minimalist approach of a small alphabet of entities with their graphical notations, which is in contrast to the large SysML alphabet symbol set over nine diagram types, it should come as no surprise that one OPM element can be typically translated to multiple types of SysML elements. For example, an OPM *process*, which is defined as an entity that transforms (generates, consumes, or changes the state of) an object can be mapped to any subset of the following SysML entities:

- *Use case* (in a *Use Case Diagram*)
- *Operation* of a block (in a *Block Definition Diagram*)

- *Action* (in an *Activity Diagram*)
- State transition *trigger* (in a *State Machine Diagram*)
- *Message* (in a *Sequence Diagram*)
- *Constraint* (in a *Parametric Diagram*)

The semantics of an OPM element is often context-sensitive. *Context-sensitivity*, which is inherent in the OPM language design, makes it possible to use a small set of symbols to express rich semantics without compromising human intuition, simplicity, and formality (Dori 2002, p. 9).

Realizing that a single "global" mapping table is not feasible, we take an aspect-view mapping approach: We partition the mapping according to the target SysML views to be generated. In other words, for each supported SysML view, there is a designated corresponding OPM-to-SysML mapping scheme. These mappings constitute the foundation for the translation algorithms from OPM to each one of the supported SysML views. Due to space limitations, the last subsection contains description of one OPM-to-SysML mapping scheme, designated for Use Case Diagram, along with the corresponding translation procedure.

At this time, out of the nine types of SysML views, we have developed mapping schemes for six diagram types: Use Case Diagram, Block Definition Diagram, Activity Diagram, State Machine Diagram, Sequence Diagram, and Requirement Diagram. While support for the remaining diagram types, namely Package Diagram, Internal Block Diagram, and Parametric Diagram, is currently missing, there is no special limitation that prevents support of these diagram types. These views were not implemented to date mainly due to practical cost-effectiveness considerations, and they can be supported using the existing architecture and design principles.

Application Framework

To construct a usable and functional OPM-to-SysML translation software application, we have chosen to use OPCAT (Dori et al. 2003) as the host environment. OPCAT is a software product that supports OPM-based system development and lifecycle management by providing an integrated development and evolution environment for conceptual modeling of complex systems. All the OPM models described and presented in this paper were also created using OPCAT. An extensible environment, OPCAT provides for convenient development of new modules via an API that enables a plug-in-based architecture, making it the platform of choice for our purpose.

Figure 1 shows an example of the main OPM-to-SysML dialog window, which appears upon invocation of the *SysML generation* operation in OPCAT. The dialog window has six boxes, one for each of the supported SysML diagram types. The user can select what SysML views to generate, and for some of the views there are several parameters that the user can set or change. The automatically-generated SysML model, which is the output of executing the OPM-to-SysML application, is created in the XMI format (OMG 2007). A standard for exchanging metadata information, XMI is widely supported by SysML (and UML) tools. Having the generated SysML model in XMI format promotes wide interoperability, as it allows importing the model into any standard-supporting SysML tool. After importing the XMI model to a SysML supporting tool, the existing capabilities of the tool for visualization and manipulation of the generated SysML diagrams can be used. In this research, we have used Enterprise Architect™ by Sparx Systems (2009) with MDG Technology for SysML as our SysML tool. Figure 2 is a screenshot of Enterprise Architect's project browser window after importing a sample XMI file. The automatically-created SysML model is organized here according to the type of view, one of the conventional ways of model organization.
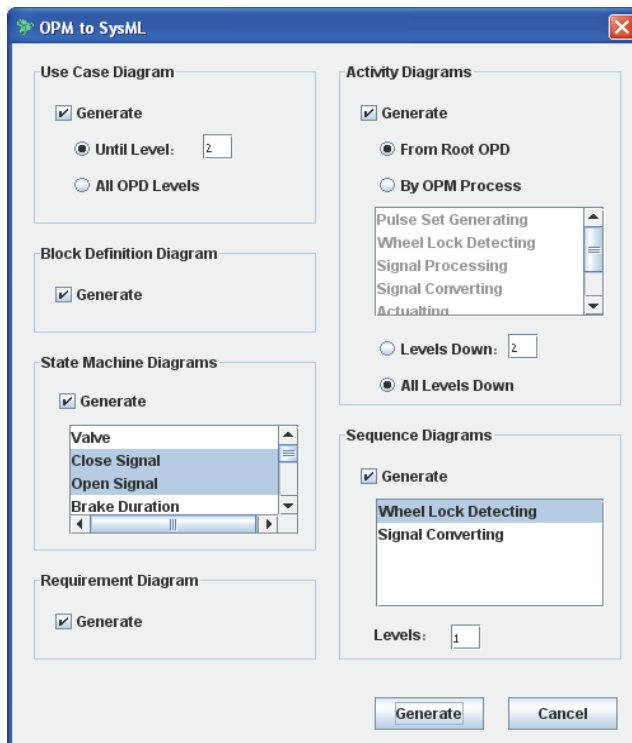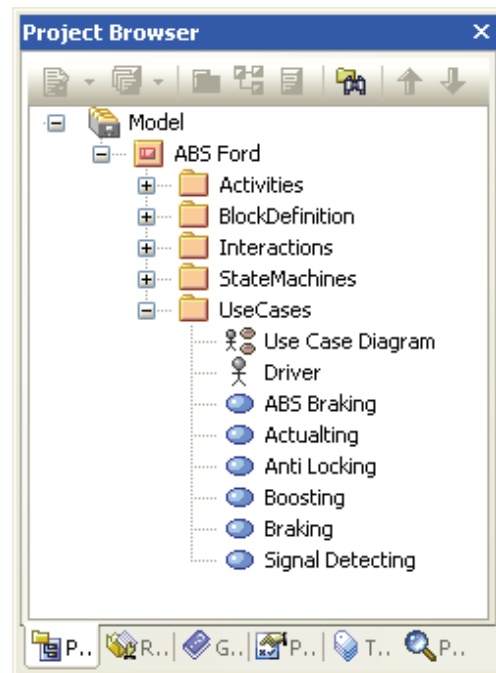
Figure 1. OPM-to-SysML main dialog window

Figure 2. Enterprise Architect's project browser showing the model elements

From the user perspective, a typical usage of the OPM-to-SysML application consists of two main stages: (1) Creation of the XMI file that contains the resulting SysML model and diagrams description. (2) Import of the XMI file into a SysML-supporting tool and further usage according to the needs. Stage (1) is executed within the OPCAT environment, while stage (2) is executed within a standard SysML tool, independently of our application. A future development we are considering is to incorporate the SysML views within an XMI-supporting version of OPCAT.

Example: Use Case Diagram

To demonstrate our approach, we describe the OPM-to-SysML mapping for the Use Case Diagram. The Use Case Diagram is intended for modeling the usage of a system, so typically the diagrams provide mainly a high-level functional view of the system and the actors. The main elements comprising the Use Case Diagram are actors and use cases (the entities) along with the relationships (links) among them. Generation of a Use Case Diagram from OPM is therefore based on *environmental objects* (the actors) and the *processes* (the use cases) linked to them. We enable generation of the resulting diagram using the first k OPD levels. The number of OPD levels (k) can be specified by the user via the interface shown in Figure 1. Typically k should be between 1 and 3, and the default value is 2. Specifying all OPD levels is also possible.

Figure 3 is an example of Use Case Diagram generation with $k=1$, where the left-hand side shows the root OPM diagram (SD), while the right-hand side contains the automatically-created Use Case Diagram. Figure 4 shows a second-level OPD from the same OPM model (on the top), and the auto-generated Use Case Diagram, this time using $k=2$ (on the bottom).
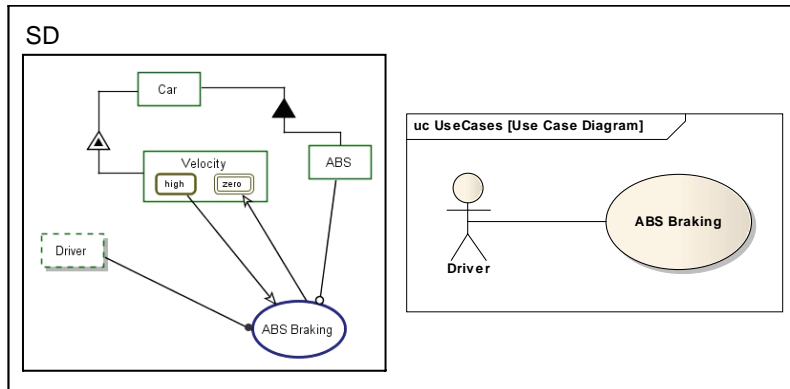
Figure 3. Example of Use Case Diagram generation (*k*=1)

Left: The original OPM System Diagram (SD, top-level diagram);
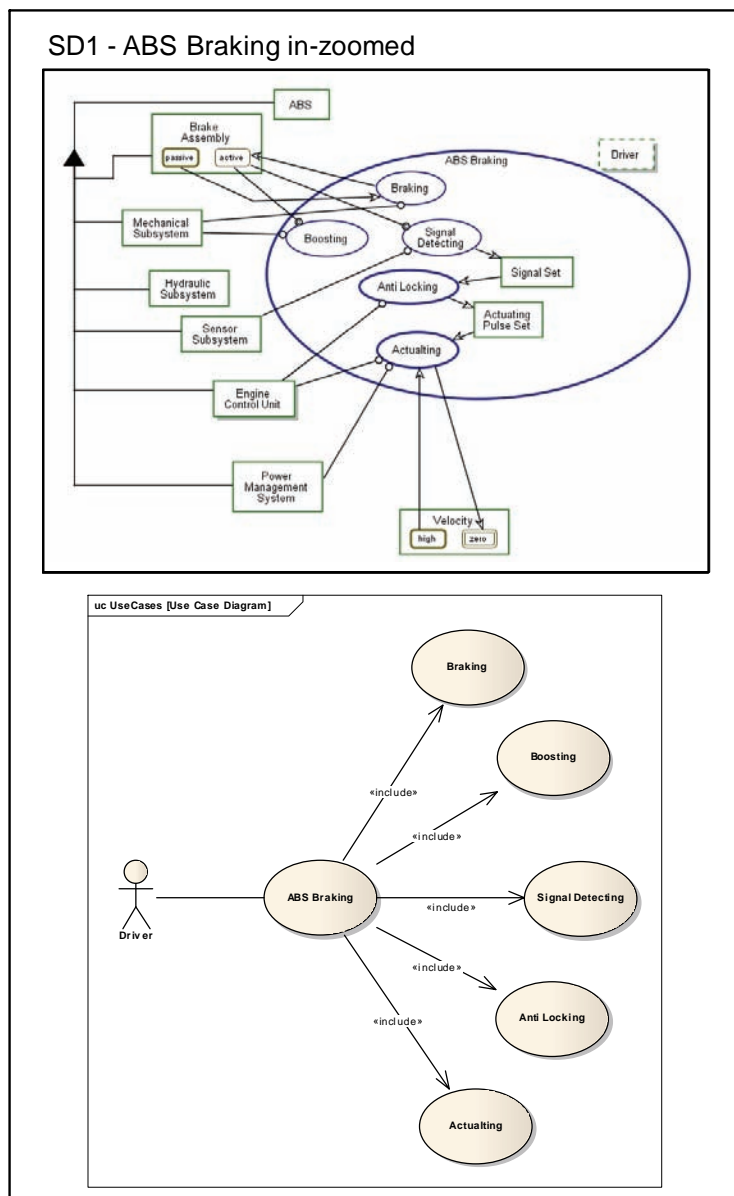Right: The auto-generated SysML Use Case Diagram
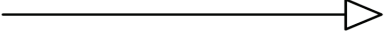
Figure 4. Example of Use Case Diagram generation (*k*=2)

Top: The original OPM diagram (SD1, one level below SD);
Bottom: The auto-generated SysML Use Case Diagram

The mapping scheme from OPM to the Use Case Diagram is summarized in Table 1.

Table 1. The OPM to Use Case Diagram mapping scheme

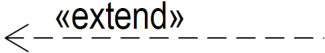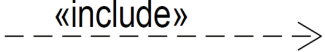| Rule #	OPM	SysML
1	**Environmental Object** Object	**Actor** ActorName
2	**Systemic Process** connected with Environmental Object (and all of his subprocesses) Object Processing	**Use Case** UseCaseName
3	**Generalization-Specialization** relation between Agent ("actor") Objects or between Processes	**Generalization** of Actors (for Objects) or Use Cases (for Processes)
4	**Exception Link**; **Invocation Link**	Use Case **«extend»** relationship «extend»
5	Process **In-zooming**; **Aggregation-Participation** relation	Use Case **«include»** relationship «include»
6	Any type of **Procedural Link** between an Agent ("actor") Object and a "use case" Process	**Association** between Actor and the corresponding Use Case

The Use Case model creation algorithm

An outline of the Use Case model creation algorithm follows (as noted, *k* represents the required OPD level):

1. Find all target use case elements (rule #2 in Table 1) within the first *k* levels. For each such element:
   1.1. Write a use case element to the XMI file.
   1.2. Find and write to the XMI file its extended use cases (rule #4), its included use cases (rule #5), and its generalized (parent) use cases (rule #3).
2. Find all target actor elements (rule #1) within the first *k* levels. For each such element write an actor element to the XMI file.
3. Find all the procedural links that connect some object *o* which was mapped to actor and some process *p* which was mapped to a use case (rule #6). For each such link *l*, object *o* and process *p*, if there is another link of the same type as *l*, which connects the same object *o* with a subprocess of *p*, do nothing. Otherwise, write an association element to the XMI file (rule #6).

The purpose of the condition in step 3 is to avoid specifying redundant associations in the XMI file; only the association at the maximum level of detail (according to k) shall appear in the resulting SysML model.

Evaluation

In order to assess and examine the effectiveness of the newly developed OPM-to-SysML algorithm and application, we performed a controlled experiment, which had two major goals: (1) Since improved system comprehension was among the potential predicted benefits, we ventured to examine whether the additional SysML diagrams that had been generated automatically with our new application, affected system model comprehension. (2) We wished to test the quality of the auto-created diagrams, mainly in terms of modeling errors and inconsistencies with the original OPM model from which they were generated.

This section is divided into six subsections as follows: In the first subsection we describe the experiment population, its background and training. The experiment design is specified in the second subsection. In the third subsection we present the experiment results, which are discussed and interpreted in the fourth subsection. In the fifth subsection we review limitations of the experiment and conclude with some insights in the last subsection.

Population Background and Training

We carried out the experiment within a lecture of the course "Enterprise Systems Modeling" taught during the 2008 Spring semester at the Technion – Israel Institute of Technology. A total of 78 students participated in the experiment. Most of the students (75) were third- or fourth-year undergraduate students in a four-year B.Sc. engineering program at the Faculty of Industrial Engineering and Management. The population also included two graduate (M.E.) students and one third-year undergraduate student at the Information Systems Engineering program.

The background of the students in systems modeling was based mainly on a mandatory prerequisite course "Specification and Analysis of Information Systems", which they had taken in a previous semester. This course covered, among other topics, system development lifecycle and two modeling techniques: OPM and UML. The course included several modeling assignments involving modeling in both OPM and UML that the students were required to submit, as well as examination questions that were given on these subjects.

The experiment was conducted in the middle of the semester. In the lectures and tutorials of the "Enterprise Systems Modeling" course prior to the experiment, several subjects related to enterprise systems modeling were taught, using OPM as a modeling language. In addition, as part of the course program, the students were introduced to SysML during a three-hour lecture that was given one week before the experiment took place. This is the only SysML training the students got.

Experiment Design

The experiment took place during a single lecture session, in the usual place and time of the course. The students were told beforehand about their participation in an experiment, but they were neither told about the goals of the experiment nor about the subjects of this research. The experiment was executed in an examination-like setting: the students got printed forms with questions, and were asked to write their answers and return the forms. In order to motivate the students to participate in the experiment and to answer the questions attentively, they were granted up to two credit bonus points that were added to their final 0-100 scale course grade. The students were told that their answers would be checked and graded, and that the number of granted bonus points will be determined according to their total score in the experiment. The

students were allowed to use any written material, including class notes and lecture slides. All the material given to the students was in English and they were told that they could choose to answer in either English or Hebrew, the mother-tongue of the most of them. In the beginning of the experiment, the class was divided arbitrarily into two groups, A and B, according to the first letter of the students' first name. Group A consisted of 36 students, while Group B had 42 students.

Model specifications of two systems provided the basis for the experiment. One was of a private residence standard dishwasher, and the other—of a computed tomography medical imaging device (CT). The systems were modeled in OPM by the first author of this paper and were validated by two other experts. For each of the systems, we used the OPM-to-SysML tool to generate several SysML diagrams. We prepared two printed versions of each of these two systems. One version contained only the original OPM diagrams, while the other version contained the same OPM diagrams along with the automatically-generated SysML diagrams added. The model of the Dishwasher system contained seven OPM diagrams, while the one of the CT system included nine. Four types of SysML diagrams were used in the experiment: Use Case, Block Definition, Activity and State Machine. Ten and eight SysML diagrams were automatically generated for the Dishwasher and the CT systems, respectively.

For each system we composed eight open comprehension questions that covered different dynamics and structure aspects of the systems and could be answered by consulting the model specification. The students were given the printed model specifications in either the OPM-only version or in the combined OPM-and-SysML version, and were asked to answer the comprehension questions. The allocation of the systems and the modeling versions of the students' groups are described in Table 2. Overall, every student received both the Dishwasher and the CT systems, one in an OPM-only version and the other in an OPM-and-SysML version. As noted, the students were not told about our research subjects. In particular, they did not know that the SysML diagrams were generated automatically from the OPM model.

Table 2. Allocation of systems and modeling specifications to groups

System	Group A	Group B
Dishwasher	OPM only	OPM+SysML combined
CT	OPM+SysML combined	OPM only

The experiment was conducted in a multi-stage manner as follows: In the first stage, which lasted five minutes, the students received the printed model specification only. They were asked to study the specification, and were advised to try and get an overall picture of the system by reviewing a large number of diagrams.

In the next stage, the students were given two additional forms while holding on to the system specification they had received. The first form contained the comprehension questions, which were identical for the two model specification versions of the same system (OPM-only and OPM-and-SysML). In the second form, the students were asked to specify any inconsistencies, errors or contradictions they find among the model diagrams, regardless of the diagrams' modeling language. The students were given 20 minutes to complete both the comprehension and error detection tasks. After 20 minutes the forms were collected and the two stages were repeated for the same amount of time with the other combination of system and modeling specification. At the beginning of the experiment, the students responded to a short "demographic" questionnaire about their faculty, semester, and background in modeling languages. This data is summarized in the previous subsection. At the end of the experiment we gave the students another questionnaire, asking them to express their opinion about the contribution of the SysML diagrams to their model understanding. All the questionnaires were

checked by the first author of this paper. The comprehension questions were checked and scored according to a consistent grading policy.

Our null hypotheses (H_0) were that for both the Dishwasher and the CT systems, there is no difference in the comprehension level between the OPM-only model specification and the combined OPM-and-SysML model specification.

Results

Each comprehension question could score a maximum of 5 point, totaling 40 points for each system. Incomplete answers, or answers with missing elements, scored less according to the detailed grading policy. Figure 5 and Figure 6 show students' average score for each question and each group in the Dishwasher and CT systems, respectively. In both cases, almost all the questions (7 out of 8) scored higher when the given OPM system model was enriched with the auto-generated SysML diagrams.

Dishwasher Comprehension Questions Scores

	1	2	3	4	5	6	7	8
Group A (OPM only)	3.89	3.72	2.53	2.94	2.56	0.72	1.22	1.50
Group B (OPM+SysML)	4.02	3.95	1.76	3.38	4.00	1.38	2.43	3.67
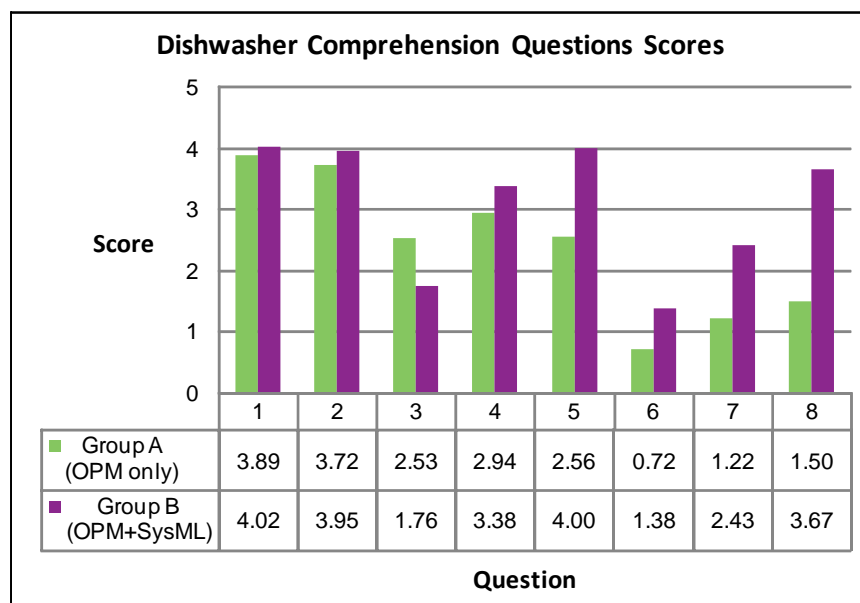
Figure 5. Average scores of the Dishwasher comprehension questions for each group

CT Comprehension Questions Scores

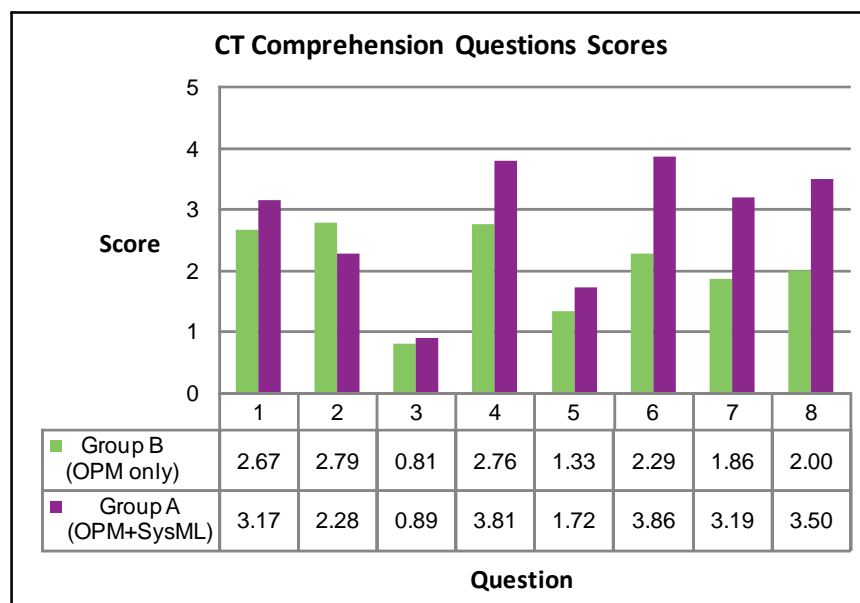|	1	2	3	4	5	6	7	8
Group B (OPM only)	2.67	2.79	0.81	2.76	1.33	2.29	1.86	2.00
Group A (OPM+SysML)	3.17	2.28	0.89	3.81	1.72	3.86	3.19	3.50

Figure 6. Average scores of the CT comprehension questions for each group

We analyzed each of the two systems separately by taking the sum of all the eight comprehension questions. To determine whether there is a significant difference in the overall comprehension level between the OPM-only model specification and the OPM-and-SysML one we employed independent two-tailed t-test. Since we could not assume a-priori equality in the variance of the scores between the two groups, we used t-test assuming unequal variances (heteroscedastic). The results are summarized in Table 3.

Table 3. Results of model comprehension

System	OPM-only		OPM-and-SysML		t-statistic (unequal variances)	p-value (two-tailed)
	Average	Variance	Average	Variance		
Dishwasher	18.97	51.57	24.60	33.61	-3.763	< 0.001
CT	16.50	51.67	22.42	52.02	-3.617	< 0.001

Testing the null hypothesis that there is no difference in comprehension level between the OPM-only model and the combined OPM-and-SysML model, against the two-sided alternative that there is a difference, we reject the null hypothesis in both the Dishwasher and the CT system cases. For both systems, the students' answers to comprehension questions relating the combined OPM-and-SysML models were significantly better than answers relating to OPM-only models.

Our purpose in asking the students to specify errors, inconsistencies or contradictions they encountered among the different model diagrams for each system was to evaluate the quality of the automatic SysML diagram generation. Since the students did not know the purpose of the experiment, nor that the SysML diagrams were automatically generated from the OPM model, they were asked to relate to all the diagrams they were given, regardless of the modeling language. In other words, they were asked to specify inconsistencies between two different OPM diagrams or between two different SysML diagrams or between one OPM diagram and one SysML diagram.

Some of the students did not find any errors. The answers of the others were carefully examined. Most of the problems reported by the students were considered as "false positives", meaning that the students were wrong and the specified issues were not errors or inconsistencies at all. Notwithstanding, the students did find some true issues that related only to the OPM diagrams. All these issues were minor and had no effect whatsoever on the system understanding. The students did not find issues related to the SysML diagrams. Specifically, errors, inconsistencies or contradictions were found neither between an OPM diagram and a SysML diagram nor between one SysML diagram type and another.

Analysis of the students' answers regarding the usefulness of the SysML diagrams shows that 74% of the participants (58 out of 78) indicated that the SysML diagrams were helpful in at least one aspect. In case they found the SysML diagrams helpful, the students were also asked to specify what types of diagrams helped them. They could have specified more than one diagram type. The results are summarized in Table 4.

Table 4. Summary of SysML diagrams helpfulness in the experiment

Diagram Type	Count	Percentage (of 58)
Use Case	9	15.5%
Block Definition	27	46.6%
Activity	22	37.9%
State Machine	23	39.7%

Interpretation and Discussion

The results show a significant difference in the overall level of comprehension between the specification that contained only OPM diagrams and the specification that included in addition automatically-generated SysML diagrams, in favor of the latter. Significantly higher scores were obtained with the additional SysML diagrams for both the Dishwasher and the CT system models.

The total average score in the CT and the Dishwasher case was higher by 29% and 36%, respectively. The average absolute scores were higher in the Dishwasher case, presumably since the students are more familiar with it as a household appliance. These results may suggest that the helpfulness of the additional SysML diagrams is more significant when the reader does not have prior knowledge about the system under study. This is a conjecture that agrees with the experiment results, but it was not tested per se. It seems that the additional diagrams were helpful in answering some of the questions more than the others. Only one question of eight in each of the cases scored lower when the combined diagrams specification was used. The SysML diagram which seems to have the biggest added value is the Block Definition Diagram (BDD), which specifies system hierarchy and features of blocks like attributes and operations. The relatively high helpfulness of the BDD is evident from both the students' assessment of the contribution of this diagram type and from analysis of the scores in the individual questions in both systems. According to the results of the experiment, the Use Case diagram seems to have the least effect on system comprehension, and this is also in agreement with the students' assessment of the Use Case diagram type contribution. In-between are the Activity diagram and the State Machine diagram, which appear to have a positive effect, albeit not as high as the effect of the BDD.

As noted, the other goal of the experiment was to assess the quality of the OPM-to-SysML application in terms of errors and inconsistencies between the source OPM model and the generated SysML diagrams. In this regard, the students did not find any issues of errors, inconsistencies or contradictions associated with the SysML diagrams, indicating that the application has achieved a high level of model-to-model translation fidelity. The significant improvement in the level of comprehension is also a positive indicator for the quality of the application in this sense.

Limitations

As with any experiment, our experiment also has several limitations. One limitation of our experiment is that as part of our research, we both developed the OPM-to-SysML translation algorithm and application and also prepared the experiment and checked it. This might have affected the experimental outcomes. This limitation was partially mitigated by validation prior to the experiment execution of all the experiment materials (including systems models and questions) by an expert who was not directly involved in this research. The concise, clear grading policy that was used in checking the questions ensured fair and uniform grading.

Another possible limitation pertains to the relative helpfulness ranking among the different types of diagrams that was done by the students with respect to a specific set of comprehension questions. Although we tried to have the questions diverse and balanced, given a different mix of aspect-related questions might have produced different results. Hence, conclusions regarding which diagram types are more or less helpful should account for this potential limitation.

One other concern is related to the experiment population, a homogenous group of students with similar background and no significant knowledge or experience in systems modeling apart from what they learned in one or two academic courses. The scope of the experiment itself was limited and included only two systems with modest complexity. The experiment

outcomes might be different with a population of experienced analysis and design experts, or with systems of greater complexity.

Finally, we note that the students in the experiment received only the diagrammatic part of the OPM model, namely the Object-Process Diagrams, without the textual representation OPM exhibits, namely Object-Process Language (OPL). In addition, the students performed the experiment with the diagrams printed on paper, and not in front of a computer-aided modeling tool. Addition of OPL, working with a computer-based modeling tool, or a combination of the two might influence the effectiveness of the SysML diagrams compared with our findings. The SysML training of the students who participated in the experience was limited as it included only one overview lecture given one week prior the experiment. With more SysML knowledge and experience, the effectiveness of the additional SysML diagrams might have been even higher.

Conclusions

Overall, we can summarize the results of the experiment with two main conclusions. The first conclusion is that enriching OPM with additional views, specifically certain SysML diagram types, improves the understanding of a given system model. While previous studies suggested that OPM is better than UML (or one of its ancestor, OMT) in terms of comprehension (Reinhartz-Berger and Dori 2005; Peleg and Dori 2000), in our case the SysML views were used *in addition* to the OPM diagrams, not *instead* of them. This way, stakeholders involved in system development can have the best of both worlds.

The second conclusion concerns the quality of the algorithm and application we developed for automatic generation of SysML views from an OPM model. Since the SysML diagrams in the experiment were generated automatically using this application, the outcomes suggest that the application we developed is indeed effective and achieves its purpose of faithfully translating OPM to SysML. Demonstrating that it is possible to automatically create useful SysML views makes the approach of enriching OPM with additional diagrams worth pursuing. Overall, the experiment result of improved systems understanding confirms a major potential benefit of the additional SysML views that complement the OPM model.

The significant experiment results suggest having a subset of SysML views created and visualized in real time by the modeling tool. Keeping the SysML views in sync with the OPM model while creating and modifying the OPM diagrams seems to have significant added value for the modeler in the modeling process. A similar approach is taken by OPCAT with the textual representation of OPM (OPL), and from our experience it seems to be very helpful.

Summary and Future Work

In this study we have developed and applied an algorithm for automatic SysML views generation from an OPM model. For each SysML view, a mapping scheme from OPM elements to SysML elements constitutes the foundation for the OPM-to-SysML sub-algorithm and translation engine application. We demonstrated the software implementation into OPCAT, the OPM-supporting modeling tool.

In order to assess and examine the effectiveness of the newly developed OPM-to-SysML algorithm and application, we performed a controlled experiment. In the experiment we found that there is a statistically significant difference in the level of overall comprehension of given system models between models specified in OPM only and models specified in OPM together with automatically-generated SysML diagrams, in favor of the latter. No indications of inconsistencies, errors or contradictions between the original OPM model and the generated SysML diagrams were discovered. The results of the experiment confirm that the approach we took—suggesting generation of various aspect views automatically from a source

model—enhances the overall model comprehension, creating synergies between OPM and SysML that should be further explored.

This work can be extended in several directions. One possibility for future research is to develop translation in the other direction, namely from SysML to OPM. The challenges and solutions in this case seem to be different than the ones we encountered in the OPM-to-SysML case, since it requires gathering information from several model aspects (diagrams) into one unified model—a many-to-one mapping. Another possible direction is to examine the applicability of our approach and findings to other modeling languages and domains.

Finally, an interesting research direction is to combine SysML and OPM benefits by defining a new unified language derivative on the basis of both of them. Initial investigation and discussions with key people involved in the development of SysML suggest that an appropriate approach is to develop a "light" version of SysML for this purpose ("SysMLite"), which will be based on a subset of SysML and ideas from OPM, using the current SysML Activity Diagram as the basis and extending it with Block Definition Diagram capabilities to be similar to OPD, while also adding text generation capabilities. The objective is to make this new SysML variant more accessible to systems architects and other stakeholders for early conceptual design, while still enabling them to take advantage of the rich SysML vocabulary when it becomes necessary, typically at later stages of the design process. This will enable holistic modeling of system structure and behavior, ideally in a single type of diagram. We also suggest leveraging OPL, OPM's textual modality, by including it, and possibly other OPM concepts, in SysMLite. Technically, SysMLite will be specified as a SysML profile, the standard extension mechanism of UML and SysML. This profile will eventually be submitted to the OMG for adoption and standardization in response to an upcoming call.

References

Dori, D. 2002. *Object-Process Methodology: A holistic systems paradigm*. Berlin: Springer.

Dori, D., I. Reinhartz-Berger, and A. Sturm. 2003. Developing complex systems with Object-Process Methodology using OPCAT. *Lecture Notes in Computer Science* 2813:570-572. Berlin/Heidelberg: Springer.

Grobshtein, Y., and D. Dori. 2008. Evaluating aspects of systems modeling languages by example: SysML and OPM. *In Proceedings of the 18th annual international symposium of the International Council on Systems Engineering (INCOSE 2008)*. Seattle: INCOSE.

Grobshtein, Y., V. Perelman, E. Safra, and D. Dori. 2007. Systems modeling languages: OPM versus SysML. *In Proceedings of the IEEE international conference on systems engineering and modeling (ICSEM'07)* 102-109.

Object Management Group. 2007. MOF 2.0/XMI mapping, version 2.1.1: OMG available specification. OMG document number formal/2007-12-01. http://www.omg.org/docs/formal/07-12-01.pdf.

———. 2008. OMG Systems Modeling Language (OMG SysML): Version 1.1. OMG document number formal/2008-11-02. http://www.omg.org/docs/formal/08-11-02.pdf.

———. 2009. OMG Systems Modeling Language: The official OMG SysML site. http://www.omgsysml.org/.

OMG. *See* Object Management Group.

Peleg, M., and D. Dori. 2000. The model multiplicity problem: Experimenting with real-time specification methods. *IEEE Transactions on Software Engineering*, 26 (8): 742-759.

Reinhartz-Berger, I., and D. Dori. 2005. OPM vs. UML - experimenting with comprehension and construction of web application models. *Empirical Software Engineering* 10 (1): 57-80.

Sparx Systems. 2009. Company web site. <u>http://www.sparxsystems.com.au/</u>.

Biography

Yariv Grobshtein is a recent graduate of the Information Systems Engineering M.Sc. program at the Technion – Israel Institute of Technology. He holds a B.Sc. degree in Computer Science and an MBA degree, both from the Technion, and has 10 years of professional experience in software-rich systems development. His M.Sc. research subject is in the area of systems modeling languages.

Dov Dori is Visiting Professor at MIT's Engineering Systems Division (ESD). Between 2001 and 2008 he was Head of Technion's Area of Information Systems Engineering at the Faculty of Industrial Engineering and Management, and Research Affiliate at MIT. Between 1999 and 2001 he was Visiting Faculty at MIT's Sloan and ESD. Professor Dori received his B.Sc. in Industrial Engineering and Management from the Technion in 1975, M.Sc. in Operations Research from Tel Aviv University in 1981, and Ph.D. in Computer Science from Weizmann Institute of Science, Israel, in 1988. Between 1978 and 1984 he was Chief Industrial Engineer of the MERKAVA Tank Production Plant. His research interests include Model-Based Systems Engineering, Systems Development and Lifecycle Methodologies, and Information Systems Engineering. Between 1999 and 2001 he was Associate Editor of IEEE T-PAMI and is currently Associate Editor of Systems Engineering. He is author/co-editor of four books and author of over 150 publications. Prof. Dori is Fellow of the International Association for Pattern Recognition (IAPR), Senior Member of IEEE and ACM, and member of INCOSE.