Model-driven Architecture of a Maritime Surveillance System Simulator

Martin Monperrus, Benoit Long, Joël Champeau, Brigitte Hoeltzener, Gabriel Marchalot, Jean-Marc Jézéquel

Model-driven Architecture of a Maritime Surveillance System Simulator

M. Monperrus[1] B. Long[2] J. Champeau[3]
B. Hoeltzener[3] G. Marchalot[4] JM. Jézéquel[5]

1. Darmstadt University of Technology
2. French General Delegation for Ordnance (DGA)
3. European University of Brittany / ENSIETA
4. Thales Airborne Systems
5. INRIA & University of Rennes

Abstract

This article reports on an experiment to apply a model-driven approach for systems engineering in an industrial context. This experiment consisted of setting up a model-driven simulation environment for a maritime surveillance system. The simulation is fully based on three models, each conforming to a specific metamodel. We discuss the main advances given by model-driven orientated simulation for systems engineering.

1 Introduction

According to Bahill et al. [2], systems engineering is structured along a process called SIMILAR, standing for: State the problem; Investigate Alternatives; Model the system; Integrate; Launch the system; Assess performance; Re-evaluate (see Figure 1). While simulation does not explicitly appear in the SIMILAR process, it is a key phase for systems engineering [15]. To a certain extent, simulation could be seen as the union of the *Investigate alternatives* and *Model the system* activities.

At the very beginning of the life cycle of a system, i.e., at the system engineering level, simulation is a way to communicate with the customer in order to elicit requirements [10]. Simulation then allows better consistency between customer requirements and delivered products. From an engineering point of view, simulation is also a way to validate parts of the architecture, as well as the behavior of the product. Last but not least, simulation enables better cost estimation and better planning of technical and human resources.

In this article, we report on an industrial experiment we undertook to develop a model-driven simulator for systems engineering following the approach promoted by Soley [22, 12] (called Model-Driven Architecture –MDA). Beyond the use of models, that have always been in use when doing

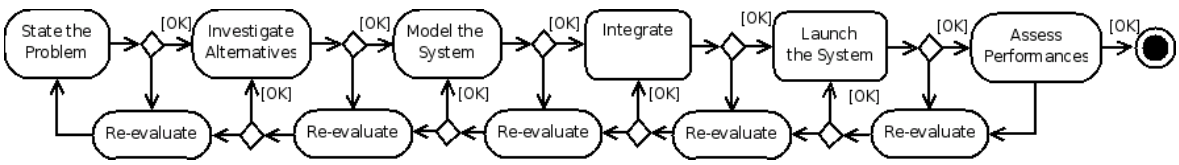

Figure 1: The SIMILAR process

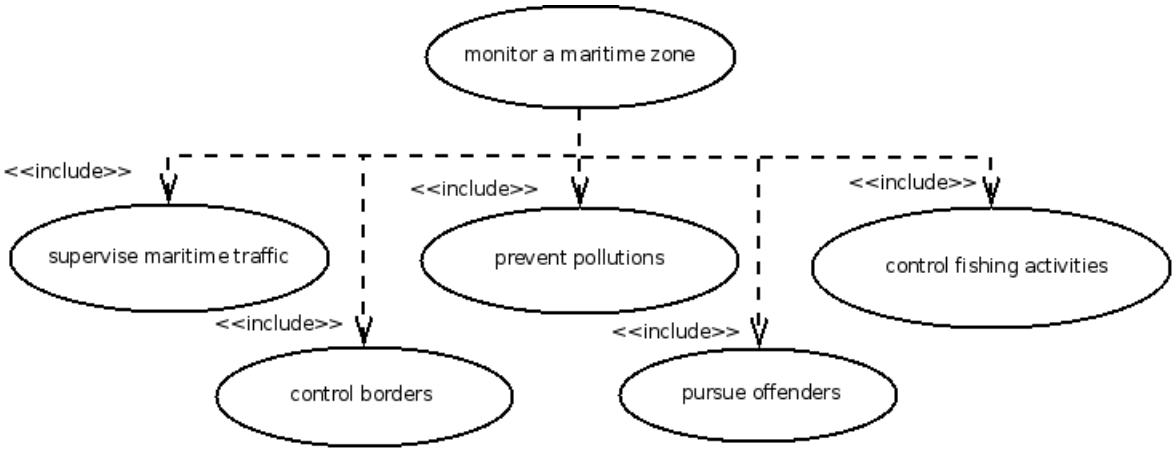

Figure 2: Use cases of a maritime surveillance system

simulations, the originality of our approach consisted in the use of model-driven principles in the sense of Soley's MDA: systematic use of explicit, ad hoc metamodels and full separation of modeling concerns. The goal of our experiment was not to bring any new functionalities over legacy simulators: we just proposed a more flexible and efficient way of building such industrial strength simulators for systems engineering. It is to be noted that the term *model-driven* used in this article is totally different from the *model-driven* expression also used for describing complex experiments involving *humans in the loop* (e.g., [21]).

The main goal of this experiment was to study the applicability and the advantages of model-driven simulation for system engineering activities. The experiment was a success. We mainly found that MDA principles enable:

- the creation of system models without being worried about implementation concerns;
- a complete independence between the specification of the system architecture, the specification of its external environment, and the analysis of the simulation results;
- the alignment of the system engineering model and the simulation model which can both be explicitly specified.

The remainder of this article is organized as follows. In section 2, we explore the design of the model-driven simulator. Then, we present the lessons learned in section 3. Finally, section 4 discusses related works and section 5 concludes this article.

2 The Design of a Model-driven Simulator

In this section, we elaborate on the application domain of the simulator we built, then we present its model-driven architecture, followed by the description of the metamodels and models we used. We conclude by exploring more in depth how we can leverage a model-driven architecture from a simulation point of view.

2.1 The Simulated Domain

The simulated systems are maritime surveillance systems [9, 10, 7]. A maritime surveillance system (MSS) is a multi-mission system. As depicted in Figure 2 in the form of a UML Use Case diagram[1],

[1]All figures of this article use the UML 2.2 specification and notation.

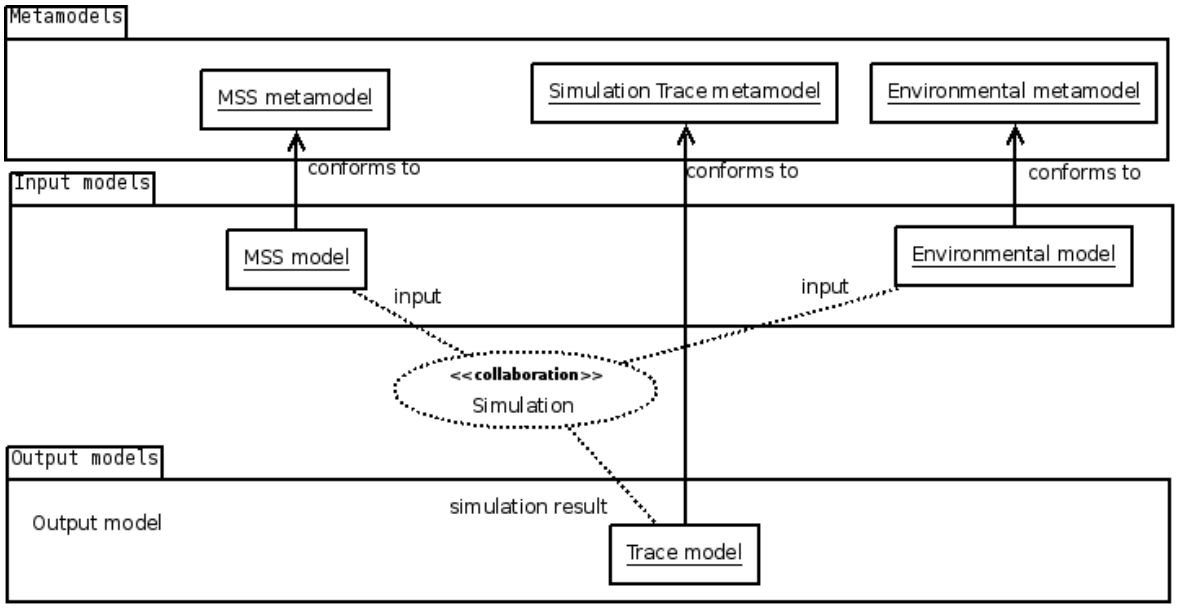

Figure 3: The MDA architecture of our simulation approach

it is intended to supervise maritime traffic, to prevent pollution at sea, to control fishing activities, to control borders and to pursue offenders. It is usually composed of an aircraft, a set of sensors, a crew and many software artifacts. The large number of possible missions, the relationships between hardware and software components and the communication between the system and other entities (base, other MSSs) lead to a complex system.

2.2 A Model-driven Simulation Approach

In Figure 3 the architecture of the model-driven simulation approach is depicted as a UML object diagram. The simulator is represented as the central UML collaboration (the dashed oval). The simulator takes as input two models: a model of maritime surveillance system and a model of the external environment, called an environmental model. It outputs a model of simulation traces. Each one of these three models is specified by an ad hoc metamodel. The contents of these metamodels are presented in the next sections.

The simulator only contains the simulation rules that represent the interactions between the system and the environment. The simulation rules define how to create trace model elements based on the system architecture and the environmental model elements. For instance, one of the simulation rules consists of creating an instance of the *PositionEvent* metaclass (defined in the trace metamodel) each time an object moves during the simulation.

This architecture is built following model-driven principles as stated by Soley [22]: 1) all data are exchanged as models specified by a metamodel 2) the domain concepts are free of implementation concerns. Since our intent is simulation and Soley's intent is source code generation, we cannot directly map the concepts of platform independent models (PIM) / platform models (PM) / platform specific models (PSM) to models of our model-driven simulation approach. However, to a certain extent, all three models are PIMs (system, environmental and trace models) since they are fully independent of the simulation platform environment and can be reused in other contexts.

This architecture has several interesting properties. All the variability of the simulator is captured into models. For instance, adding a given radar, or changing a feature of the radar to the maritime surveillance system is implemented by modifying the input MSS model only.

The MSS model and the environmental model are fully uncoupled. Therefore, we can easily create several MSS models, several environmental models and study the behavior of each system in each environment.

The graphical output of the simulator is also loosely coupled with the simulation. This property makes it easy to have many tools that analyze the simulation output, because we have one and only one interface for plugging modules into the simulator. In such a model-driven architecture, the interface is a model which is fully and consistently specified by a metamodel, the simulation trace metamodel. This point is further explored in section 2.4.

2.3 The Metamodels Used

2.3.1 Maritime Surveillance System Metamodel

The maritime surveillance system metamodel is divided into five packages, following a functional decomposition. The navigation system package contains classes whose roles are to represent routing and positioning components (hardware, software and composite). For instance, there is a GPS[2] class. The attributes of this class are the main characteristics of a GPS.

The detection package contains classes representing detection components of a MSS. For instance, the MSS model can be simulated with a classical radar, an Identification Friend and Foe system (IFF), a Forward Looking Infra Red system (FLIR). The attributes of a model of the radar include the antenna angle, the rotation speed and the impulse period.

The communication system package defines the communication types and systems that should be simulated. It ranges from internal communication between the crew members to external communications. The classes cover the communication support (e.g. very high frequency (VHF) radio telecommunication, satellite telecommunication) and communication properties (encryption, protocols).

Finally, the crew package contains classes to specify the number of crew members, their respective skills and their functions. The database package acts like a schema of the embedded database of tactical information (e.g., radar signatures).

The model used for our testing purposes involves a maritime surveillance system composed of a Falcon airplane, embedding a radar and an inertial system.

2.3.2 Environmental Metamodel

The environmental metamodel contains all the classes needed to represent a tactical situation. A model, instance of this metamodel, specifies the surveillance zone, the number and types of objects that are in the zone. For each object, a trajectory is specified, which includes the speed of the object. Furthermore, at the system engineering level, and for simulation purposes:

1. we added the concept of *Equivalent Surface Radar* (ESR) per object [10], which quantifies the object signature w.r.t. radar impulses;

2. we added the concept of weather for a given zone. During the simulation, the weather is used to compute the quality of the information given by the embedded sensors.

2.3.3 Simulation Trace Metamodel

The simulator takes two models as inputs: a MSS model and an environmental model. Then according to the semantics of the simulation, it outputs a simulation trace. Following a model-driven approach, this simulation trace is a model too, specified by a simulation metamodel.

[2]Global Positioning System

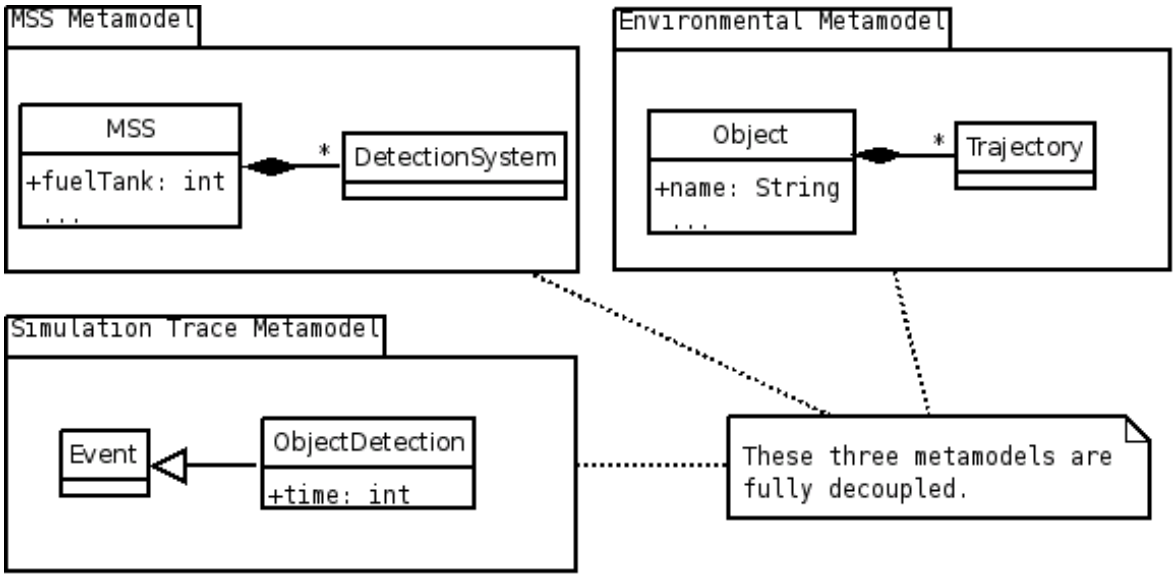

Figure 4: Excerpt from the metamodels used

The simulation traces contain all the information that are relevant to simulation at the systems engineering level:

- the position of the MSS system;
- the position of each object in the zone;
- the internal attributes of the MSS: fuel, detection field of the radar;
- the semantic events, for instance the detection and identification of the objects achieved by the system core.

Figure 4 gives an overview of these metamodels. They are specified with OMG's Meta-Object Facility (MOF [17]). They can be represented with a notation close to UML class diagrams, even if strictly speaking these metamodels are not UML class models. Models used in the simulations are instances of either of these three MOF metamodels. Put it in another way, systems engineers using the simulator have to design MSS and environmental models that conform to their corresponding metamodels.

We defined more than 50 different MSS models to be simulated. Figure 5 shows an excerpt from one of them as an object diagram. The MSS modeled has a fuel capacity of 4000L, and embeds a radar and a FLIR which are fully characterized. We generated hundreds of different environmental models based on various characteristics (e.g. number of ships, complexity of their trajectories). Note that simulation trace models were not created manually but resulted from the simulation of one MSS model along with one environmental model.

2.4 Leveraging Model-driven Architecture

In this section, we show how we use the models described above to produce as much information as possible in the simulation development.

The core of a model-driven simulator is to specify all inputs and outputs with metamodels. To a certain extent, each of these metamodels specifies an interface. This enables us to introduce a kind

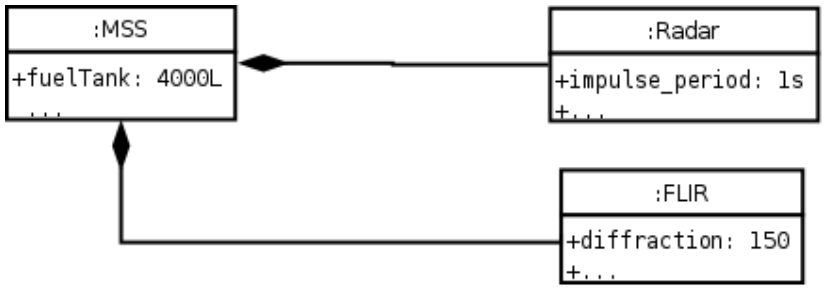

Figure 5: Excerpt from a MSS model used

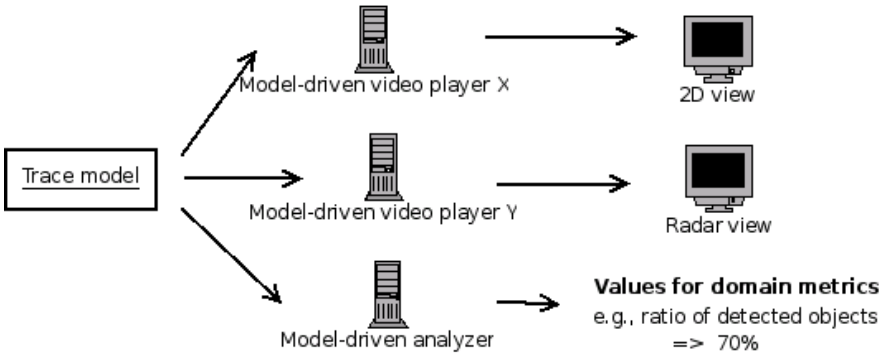

Figure 6: Usages of the simulation traces

of composition between the simulator functions. The MDA architecture of the simulator presented in figure 3 illustrates this point: no analysis is done during the simulation itself.

Our idea is to consider that any processing on the simulation output should be fully decoupled from the rest of the application. The types of processing are mainly visualization and measurement. However, there are several possible visualizations as well as several measurements. We encapsulate each of them into a simulation analysis component that takes as input a simulation trace model specified by the simulation trace metamodel. This is illustrated in Figure 6. We apply this design principle to several simulation analysis components.

The first component is a video player of the simulation events contained in the simulation trace model. It outputs the tactical view of the surveillance zone. A screenshot of this player is given in Figure 7. The main screen is a 2D representation of the zone, with a geographic map, the maritime surveillance system (the yellow dot), the objects of the zone (several ships represented as orange dots), and the detection field of the radar, which is visible as the curved yellow line in the lower right hand corner of the image. In the upper bar, there are the name of the simulation trace model used, the simulated time (the red bar), and numerical indicators of the simulation (the elapsed time in seconds and the geographical position of the MSS). In the left bar there is a list of the tactical events that occur, for instance ship detections. The remaining items are widgets to control the simulation (play/pause/stop buttons). The scale that controls the speed of the video player allows the user to play the scene at a speed different from the simulated time. This simulation component is a demonstrator to communicate with customers as well as to illustrate the impact of system design choices.

Since we already have human training processes using simulators, we developed another simulation analysis component to represent the view of a radar operator. The last simulation analysis component computes several domain-specific metric values, such as the ratio of detected objects or the ratio of identified objects. This component is generated from an abstract specification of domain

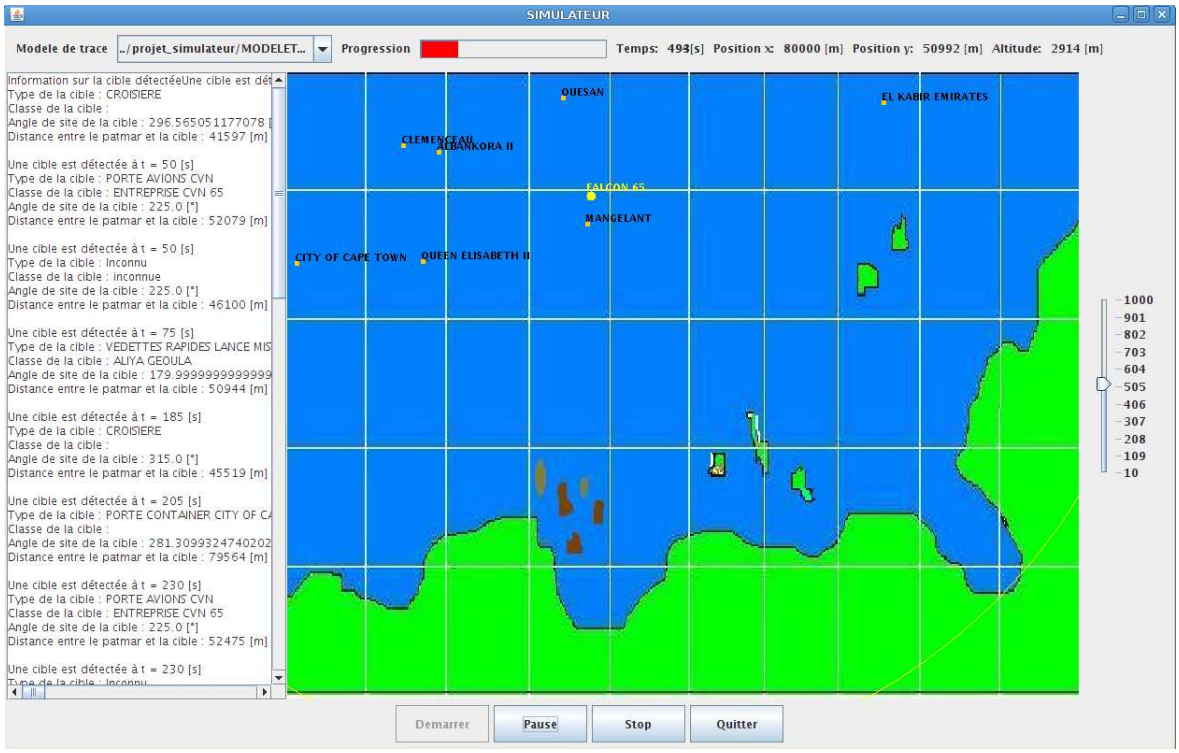

Figure 7: Screenshot of the model-driven video player

metrics, thanks to a dedicated framework developed in another project [13, 14].

Finally, since the simulation analysis components are decoupled from the simulation itself, we plan to reuse these components to analyze some traces of real missions. This will provide powerful processes for training and debriefing, as well as analyzing real missions in order to improve future versions of the simulator.

3 Lessons Learned

Reasons for success First of all, this research project on the usability of MDA at the system engineering level was a success. We found that the main technical reasons for success were: 1) the active collaboration between software engineering experts, MDA specialists and systems engineers; 2) the maturity of the MDA software libraries used (e.g. EMF [4]); 3) the productivity of the programming language and the IDE used for developing the simulator (resp. Java and Eclipse).

Productivity gain Our experiment gives some clues about a potential MDA-specific productivity gain. We obtained the prototype within six months based on the work of a junior engineer. The technologies used were the Eclipse Modeling Framework [4] coupled to Java to implement the simulator. Thanks to the EMF technology [4], the graphical model editors, used for creating MSS and environmental models, were totally generated from the corresponding metamodels. Hence, we could right away, and for free, give the system engineers, who are not necessarily computer scientists, a tool to easily express the models. This is to be compared with the legacy simulators where a specific graphical user interface had to be developed to specify the models involved in the simulation.

Separation of concerns Our model-driven simulation approach enables a real separation of concerns. The variability of a MSS simulator is made explicit and encapsulated in the system architecture, the simulated environments, the kinds of post-simulation analysis. All the variability of the models is explicitly concentrated into different metamodels. There is no dependency link between the metamodels, hence we obtained a full independence between concerns. This architecture is fully open w.r.t. system models, environment specification and analysis. For example, we are able to run the same system model with various environmental models, or to perform different analysis from the same simulation.

Early model alignment The real innovation of our experiment was the metamodeling of the domain of maritime surveillance systems. Then, the creation of models only involves domain concepts and is not polluted by implementation concerns. This was perceived as highly valuable: the system engineering model facilitates the communication between stakeholders (a kind of mental alignment) and it is totally aligned with the simulation model by construction. It seems that the maritime surveillance system model could also be automatically aligned with a design model. However, this continuous link between the simulation model and the design model has not yet been supported by a running prototype.

4 Related Work

The expression *Model-driven systems engineering* was largely coined by Wymore's book in 1993 [24]. The main idea was to introduce formalized artifacts during systems engineering activities. Wymore's proposal relies on discrete state systems.

At the end of the same decade, the general purpose *Unified Modeling Language* (UML) [18] was mature enough to be used for systems engineering. Ogren explored the usage of UML for model-driven systems engineering [16], as well as Braun et al. [3]. The idea is to leverage the rich semantics of UML to express systems engineering concepts.

Wheeler and Brooks [23] also used the UML for systems engineering but tailored it with a UML profile (UML Profile for Schedulability, Performance and Time). They described a full methodology associating several UML-driven tools (*Rhapsody*, *RapidRMA*, *OPNET*). Along the same line, Axelsson explored [1] systems engineering using a continuous-time extension of the UML.

More recently, Estefan published [6] a survey of model-driven systems engineering methodologies. He reviewed Harmony SE, INCOSE OOSEM, OPM, RUP SE, Vitech methodology and a NASA in-house method called State Analysis. He concludes that *the advantage of using industry standard visual modeling languages over vendor-specific modeling languages is clear and does not warrant debate.*

While aforementioned articles discussed general methodologies, our article relates a concrete experiment on using models for systems engineering and describes a model-driven architecture for operationalizing the system semantics.

However, there are other relevant works that deal with model-driven experiments for systems engineering. Rao et al. reported on a concrete experiment using a model-driven approach [20]. They use an extension of UML, SysML [19], to model a complex system for monitoring and collecting information related to Earth's resources. Along the same line, Haley et al. [8] presented an application of SysML to systems of systems simulations and Martin described [11] an entity-relationships (ER [5]) based architecture for observing systems.

Conversely, we do not use general purpose modeling languages, but OMG's MOF based domain-specific metamodels [17]. To the best of our knowledge, our article is the first to present the use of such domain specific metamodels for systems engineering.

5 Conclusion

While there may be a kind of consensus around the benefits of model-driven systems engineering, there are many competing formalisms (e.g., UML, SysML, domain-specific metamodels). To a certain extent, there is a lack of published experiments that could contribute to a deeper understanding of each formalism.

In this article, we presented our experiment to apply Model-Driven Architecture [22] at the systems engineering level. We discussed a model-driven simulation for maritime surveillance systems. Every object of the simulation is structured by a domain-specific metamodel: the system architecture, the environment and the simulation traces. The model-driven architecture is rooted into previous works that advocate the use of models for systems engineering (cf. [6]).

Our experiment was a success. It is part of a set of other experiments conducted within Thales to apply model-driven methods and techniques to systems engineering. The next experiment will consider the domain metamodel as the specification of a domain specific modeling language (DSML). We will also explore ways to leverage the metamodel with a dedicated human-machine interface that supports an intuitive graphical syntax for models. Thus, the domain experts would have an intuitive tool to explore the architectural solution space during the systems engineering process.

References

[1] J. Axelsson. Model based systems engineering using a continuous-time extension of the unified modeling language (UML). *Systems Engineering*, 5:165–179, 2002.

[2] A. T. Bahill and B. Gissing. Re-evaluating systems engineering concepts using systems thinking. *IEEE Transactions on Systems, Man, and Cybernetics*, 28(4):516–527, Nov 1998.

[3] P. Braun and M. Rappl. Model based systems engineering - a unified approach using UML. In *Proceedings of the 2nd European Systems Engineering Conference EUSEC'2000*, 2000.

[4] F. Budinsky, D. Steinberg, E. Merks, R. Ellersick, and T. J. Grose. *Eclipse Modeling Framework*. Addison-Wesley, 2004.

[5] P. P. Chen. The entity-relationship model - toward a unified view of data. *ACM Trans. Database Syst.*, 1(1):9–36, 1976.

[6] J. A. Estefan. Survey of model-based systems engineering (MBSE) methodologies. Technical report, INCOSE MBSE Focus Group, 2007.

[7] H. Graber and J. Curlander. Oceanview: A maritime surveillance system. In *Proceedings of the SEASAR'2006*, 2006.

[8] T. Haley and S. Friedenthal. Assessing the application of SysML to systems of systems simulations. In *Proceedings of 2008 Spring Simulation Interoperability Workshop (SIW'2008)*, 2008.

[9] A. Ince and E. Topuz. The design and computer simulation of a maritime surveillance system. In *Proceedings of Radar'97*. IEE, 1997.

[10] A. Ince, E. Topuz, E. Panayirci, and C. Isik. *Principles of Integrated Maritime Surveillance Systems*, volume 527. Kluwer Academic Publishers, 2000.

[11] J. N. Martin. On the use of knowledge modeling tools and techniques to characterize the NOAA observing system architecture. In *Proceedings of INCOSE'2003*, 2003.

[12] J. Miller and J. Mukerji. Mda guide version 1.0.1. Technical report, Object Management Group (OMG), 2003.

[13] M. Monperrus, F. Jaozafy, G. Marchalot, J. Champeau, B. Hoeltzener, and J.-M. Jézéquel. Model-driven simulation of a maritime surveillance system. In *Proceedings of the 4th European Conference on Model Driven Architecture Foundations and Applications (ECMDA'2008)*, 2008.

[14] M. Monperrus, J.-M. Jézéquel, J. Champeau, and B. Hoeltzener. A model-driven measurement approach. In *Proceedings of the ACM/IEEE 11th International Conference on Model Driven Engineering Languages and Systems (MODELS'2008)*, 2008.

[15] NASA. Nasa systems engineering handbook. Technical report, NASA, 2007.

[16] I. Ogren. On principles for model-based systems engineering. *Systems Engineering*, 3:38–49, 2000.

[17] OMG. MOF 2.0 specification. Technical report, Object Management Group, 2004.

[18] OMG. UML 2.0 superstructure. Technical report, Object Management Group, 2004.

[19] OMG. System modeling language specification v1.1. Technical report, Object Management Group, 2008.

[20] M. Rao, S. Ramakrishnan, and C. Dagli. Modeling and simulation of net centric system of systems using systems modeling language and colored petri-nets: A demonstration using the global earth observation system of systems. *Systems Engineering*, 11:203–220, 2008.

[21] A. P. Sage. Simulation and model driven experimentation in systems engineering. *Systems Engineering*, 2:57–61, 1999.

[22] R. Soley. Model driven architecture. Technical report, Object Management Group, 2000.

[23] T. M. Wheeler and M. D. Brooks. Experiences in applying architecture-centric model-based system engineering to large-scale, distributed, real-time systems. Technical report, Mitre Corp., 2007.

[24] A. W. Wymore. *Model-Based Systems Engineering*. CRC Press, 1993.