# Model-Driven Engineering of Maritime Systems

Adrian Rutle[1] and Hans Georg Schaathun[1]

Høgskolen i Ålesund, postboks 1517, 6025 Ålesund
{adru,hasc}@hials.no

## 1 Introduction

It is well-known that many features are moving from hardware to software, which is thus playing an increasing role in most engineering systems and other applications, at an increasing share of the cost. Many computer scientists have long recognised that software complexity is increasing much more rapidly than the improvements on programming languages and the methods to manage the complexity. Where software, measured in lines of code, is estimated to grow at a factor of ten per decade, very little has happened to programming languages since the introduction of object orientation around 1970.

In this project we are considering challenges and possible solutions within software for the maritime sector, in particular the development of training simulators for advanced marine operations. In the design of such training simulators, we see two major software challenges. Firstly, systems are composite and depend on the collaboration of programmers, electronic engineers, mechanical engineers, mathematicians and others. Each discipline will have to contribute to the code, but common-place languages of the day require a programming specialism to understand. Raising the level of abstraction, programming in a language which is closer to the shared mathematical syntax of the disciplines, we would reduce the risks of miscommunication, ease validation, and make the process less error-prone.

The second challenge is reuse of software components. The maritime industry is very fragmented, and every system will depend on modules from different vendors, each with their own favoured platform. Considering training simulators, we may need a propeller model, a hull model, and an ocean model from three different vendors. With different OS, language and API specification we have a problem. Again, modelling at a higher level of abstraction, a platform independent level, it should be easier to automate code generation for the platform of choice.

Our approach to handle these challenges is based on model-driven engineering (MDE), a development methodology where models are considered the first class artifacts in the development process. In MDE we shift the focus from code – the most important artefact in traditional development processes – to abstract models. These models are used in automated transformations to generate executable code. In this way, we achieve platform and architecture independence, contributing to reusable components that are easier to specify and to compose. In particular, we develop a domain specific modelling language (DSML) which can be used by various domain experts with different expertise to specify various components of the system, then we define connectors between these components based on their structural and behavioural properties.

## 2 Training Simulator for Marine Operations

A *system* is an object or collection of objects where we want to study the properties [2]. It may be a building, a bridge, an aircraft, a ship, or a piece of software. A *model* of a system is (also) a system, representing the original system. Models may be abstracted or simplified to focus on particular perspectives that are relevant to specific experiments or analyses, thus leaving out details which may be less relevant. Experiments performed on a model representing

the original system called *simulation*, and serve to demonstrate the dynamic behaviour of the system. A simulator model can *describe* an existing ship system, and at the same time *prescribe* a software system to simulate the behaviour of the ship system.

Now we outline a simplified ship simulator model. The model consists of the following main components: Hull, Controller, Thruster Control, Actuator and Propulsor (see Figure 1). Of these components, Hull and Propulsor are models of physical systems, while Actuator is a model of the software used to compensate for the latency in real systems in order to make the simulation more realistic. Controller may be a model of software (e.g., Dynamic Positioning System) or of physical system (e.g., Joystick). The Thruster Control is a model of the software which communicates with Controller and Actuator.

Figure 1 shows the ship simulator model specified in the Eclipse modeling Framework (EMF) [5]. The simulator software can be generated from this model; e.g., at any time, we can query the position of the ship, its speed and direction. In addition, we can perform queries on Actuator and Propulsor to retrieve information about these components' states.

The internal behaviour of the physical components are specified in Modelica [2]. That is, the structural model is a skeleton which can be configured with different behaviour models. The components which communicate with Modelica models, for example, `PropulsorPhysicalModel` and `HullPhysicalModel`, delegate method calls from `Propulsor` and `Hull`, respectively, to the Modelica models representing these physical systems. Compatibility between the Modelica models and the rest of the system is facilitated by a common metamodel for physical units, including the *Système International* (SI) which is already supported by Modelcica as a subset. Some of these types are shown in the lower right hand part of the figure.

Building training simulators that are suitable for training for advanced marine operations requires accurate modelling of the physical components. These components are usually modelled by different vendors. Connectors between these components are defined to represent how a

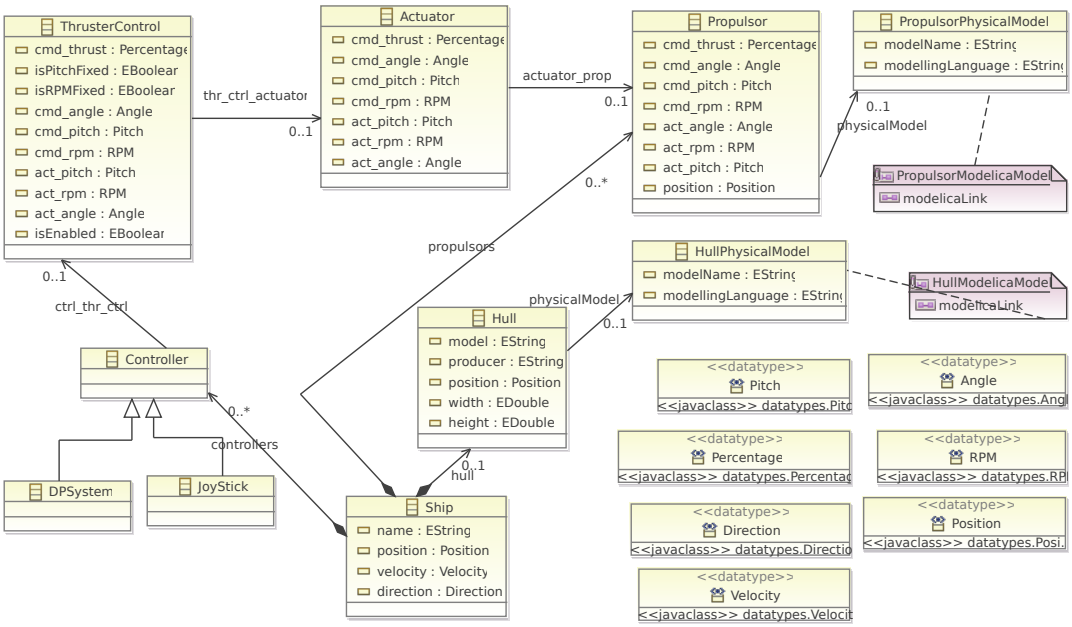


Figure 1: Ship model in EMF

component influences and is influenced by the overall system (or other components). In order to define these connectors, we could connect e.g. propellers and hulls in Modelica. However, this would lead to close coupling of these components making reuse a challenge. This would also require some knowledge in Modelica language and the underlying mathematics for the physical components. Therefore, we define these links at the model level, as in Figure 1. We know in advance how the propellers, engines, gears, etc. are connected to the hull to construct a ship, and use this information to automatically generate code for connecting the Modelica models that represent these components. The end result is that both the structural ship model and the behavioural Modelica models will be compiled to executable code.

# 3 Related and Future Work

Graphical Modelica Modelling Language (ModelicaML) is a UML profile for Modelica which enables an integrated modelling and simulation of system requirements and design [4]. ModelicaML combines UML and SysML's standardized graphical notations for system modelling with Modelica's strength in modelling and simulation of physical systems. It makes possible to create executable specification and analysis models that can be simulated in discrete and continuous time. SysML4Modelica is another UML profile that is designed to standardize the ralationships between OMG's technologies and Modelica [3]. This profile can be used to annotate SysML elements with Modelica specific information. SysML4Modelica comes with a transformation – SysML-Modelica – that generates Modelica code from SysML design models. These profiles and the transformation SysML-Modelica is all based on the OMG's well-known technologies.

The Modelica community has specified the Functional Mock-up Interface (FMI) [1] to enable model exchange and deployment across tools. This API is C-oriented (as Modelica otherwise,) thus some machine architecture, operating system and language dependence is expected.

Currently we are investigating the usability of the DSML for specification of single systems such as a ship or a crane. In future, the DSML will be extended to enable specification of training scenarios involving a various number of ships, cranes, winches, etc. Thus, we will generalise the approach from building models by connecting different components to building scenarios by connecting various models.

In this article we have outlined a solution for the development of maritime training simulators based on the MDE. The approach facilitates the development and composition of software components on the model level, and later to generate executable code that can be simulated in a training environment. We have specified a model for ship with links to Modelica models describing the behaviour of the physical components.

# References

[1] T. Blochwitz et al. The functional mockup interface for tool independent exchange of simulation models. In C. Clauß, editor, *Proc. of the 8th Int'l Modelica Conference*, pages 105–114. Linköping University Electronic Press, March 2011.

[2] P. Fritzson. *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1.* Wiley-IEEE Press, 2004.

[3] C. Paredis et al. An overview of the sysml-modelica transformation specification. In *2010 INCOSE International Symposium*, July 2010.

[4] W. Schamai, P. Fritzson, C. Paredis, and A. Pop. Towards unified system modeling and simulation with modelicaml: Modeling of executable behavior using graphical notations. In *Proc. of the 7th Int'l Modelica Conference*, pages 612–621. Linköping University Electronic Press, 2009.

[5] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks. *EMF: Eclipse Modeling Framework 2.0 ($2^{nd}$ Edition).* Addison-Wesley Professional, 2008.