Abstract. The large number of simulation languages and frameworks both in academia and industry gives rise to the problem of sharing, reusing and maintaining simulation models. The Model Driven Engineering (MDE) approach has been successfully established in software engineering for facilitating software asset management and multi-platform support. The principles of MDE can be also employed in simulation engineering. This is in line with the current tendency in the modeling and simulation area to learn from software engineering tools and techniques. From the perspective of MDE, simulation models can be considered at three different abstraction levels: the level of domain modeling, the level of (platform-independent) design modeling, and the level of (platform-specific) implementation modeling. The main idea of MDE is that 1) the design model model is derived from the domain model, 2) the implementation models are derived from the design model, and 3) executable code is generated from the implementation models.

1 Introduction

Model-driven engineering (MDE) is a software engineering approach that focuses on developing models rather than code, which can be generated automatically from computationally complete models. The Object Management Group (OMG) has proposed the Model-Driven Architecture (MDA)\footnote{MDA - http://www.omg.org/mda/}, as an MDE approach based on the fundamental idea that the chain of modeling goes from a conceptual domain model (called computation-independent model (CIM) in MDA) via a platform-independent design model (PIM) to one or more platform-specific implementation models (PSM) (one for each target technology platform), which can be directly mapped to code.

Since computer simulations are a particular class of software programs, Simulation Engineering can be considered a special case of Software Engineering. Like software engineering, simulation engineering can benefit a lot from developing conceptual domain models as the basis of platform-independent simulation models that are finally turned into executable simulation programs using some technology platform.
The standard view in the simulation literature is that a simulation model can be expressed either in a general purpose programming language or in a specialized simulation language. This means, that the term model is used rather loosely both for low-level computer programs and for higher-level executable specifications. There is often no distinction between a conceptual/logical system model (expressed either as a non-executable conceptual model or as an executable specification in a high-level simulation language) and its implementation in some target technology platform.

Clearly, as in software engineering, such a distinction would be important for several reasons: as opposed to a low-level computer program, a high-level simulation model would be more comprehensible and easier to communicate, share, reuse, maintain and evolve, while it could still be transformed into any platform-specific implementation code. The main purpose of working with high-level simulation models is to facilitate the reuse, the publication and the interchange of simulation models between different systems and tools.

Such kind of high-level simulation models always suppose a high-level, meta-language in which their syntax should be described. For example, the "eXtended Markup Language", XML\(^2\), is clearly a mature software engineering standard but, it is also largely used in the modeling and simulation area as a data format (see, for example, the Brain framework\(^1\) or the Modelling4All framework\(^3\) or OMG’s SysML\(^4\) emerging interchange language). Notice that even standardisation communities such as OMG\(^4\) focus their work on providing specification languages for MDA’s CIM and PIM layers of simulation models in order to comply with main MDE principles such as reuse and inter-operability through standardisation procedure.

The OMG community dominated by software suppliers and consultants does not produce software products but delivers only specifications (also called OMG Standards). In the OMG’s area of software engineering there are general-purpose standards which are very attractive to the modeling and simulation community and sometimes considered as part of the methodologies and tools of the modeling and simulation area (e.g. UML\(^5\) [the Unified Modelling Language], XMI\(^6\) [the XML Metadata Interchange], SysML [the Systems Modelling Language] or BPMN\(^7\) [Business Process Modeling Notation]). In the context of OMG’s MDA, MetaObject Facility (MOF\(^8\)) represents the metamodeling language. As ([2] - page 20) sustains, the actual literature about metamodeling induces a confusion about its real meaning. Standards such as UML and OMG’s MOF provide metamodels that claim to define the standard, but they only focus on the abstract syntax (i.e. the vocabulary). Though it can not provide an executable model,\(^9\)

---

\(^2\) XML - http://www.w3.org/XML/

\(^3\) SysML - http://www.omg.sysml.org/

\(^4\) OMG - http://www.omg.org

\(^5\) UML - http://www.uml.org

\(^6\) XMI - http://www.omg.org/spec/XMI/2.1.1/

\(^7\) BPMN 2.0 beta - http://www.bpmn.org/

\(^8\) MOF - http://www.omg.org/mof/

\(^9\) References and links are proprietary and refer to the original document.
OMG’s UML is widely used in software engineering to define a rigorous and precise specification to which any executable program language must conform.

Moreover, all these OMG Standards mentioned below address somehow the topic of conceptual modeling, but they are in fact only some data formats (expressed either textually as XMI and SysML interchange languages or graphically such as MOF/UML or BPMN diagrams).

In conclusion, OMG’s MDA and its supporting process MDE seem to be adapted to the modeling and simulation activity and should be in use in simulation system developments.

Recognizing some limitations of the current approaches, we show how a basic discrete event, XML-based simulation language situated at the PIM level can be enabled to develop simulation scenarios, which are later translated into PSM languages (e.g. Java\(^9\), Javascript\(^{10}\), OpenSimulator \([6]\)) and further executed.

In this regard, we introduce and discuss in the next section an open source DES framework, called ER/AOR simulation, which is an ontologically well-founded agent-based DES framework with a high-level rule-based simulation language and an abstract simulator architecture and execution model available from http://www.AOR-Simulation.org. Further, we discuss the works of other similar simulation frameworks/environments in the context of MDA model-driven approach.

The ER/AOR Simulation framework was proposed in \([5]\) and actually is developed at the Chair Of Internet-Technology, Cottbus under the supervision of prof. Gerd Wagner. An ER/AOR simulation model is conceptualized as a general pattern for describing a class of simulation scenarios that can be instantiated from the template simulation model based on different input parameters. With other words, The simulation model describes the behavior and entities of the system under consideration, while the simulations scenarios are experiments performed on the model. A simulation scenario essentially consists of a simulation model and an initial state. The simulation scenario is expressed with the help of the high-level, XML-based ER/AOR Simulation Language (ERSL/AORSL). It can be further translated using eXtensible Stylesheet Language Transformation scripts (XSLT) into: (1) Java source code, compiled to Java byte code and finally executed, or (2) JavaScript code and further executed on any device that supports a Web browser.

We can situate the ER/AOR simulation models at the level of MDA PIM’s level of abstraction, due to the fact that they can define simulation scenarios that are further translated into executable code such as Java and JavaScript and run on any system enabled with JVM, and on any device enhanced with a Web browser, respectively.

Another significant academia work which focuses on providing simulation applications for MDA’s PIM level via web browsers is the Modelling4All project\(^{11}\). The aim of this project is building a web-based tool for constructing, running, vi-

---


\(^{10}\) Simulario - http://www.simulario.de/

\(^{11}\) Modeling4All - http://modelling4all.org
ualizing, analyzing, and sharing agent-based models. The simulation models are constructed by composing pre-built modular components which are called microbehaviors. The BehaviorComposer tool is responsible for assembling the simulation models on different simulation platforms such as NetLogo or Repast. Another further direction envisioned is producing executable code (running scripts) for the multiuser virtual environment Second Life.

The SysML XML-based language is proposed by the OMG as a simulation interchange language. Few interchange works which involve SysML language exist on literature. In [4] a model-to-model transformation approach is proposed.

Our goal in this paper is to explain the merits and great potential of a model-driven approach also in the area of modeling and simulation.

References