Journal of
Information Technology & Software Engineering

Daniluk, J Inform Tech Softw Eng 2012, 2:3
DOI: 10.4172/2165-7866.1000e105

Editorial Open Access

Modeling in the Process of Constructing the Software for Scientific Purposes

Andrzej Daniluk*

Department of Applied Computer Science, Institute of Computer Science, Maria Curie-Skłodowska University, Poland

Creating and implementing a design solving a chosen scientific problem comes down to the correct solution of a given problem from the standpoint of the scientific theory that we use, the architectural design and abstraction of the system in the modeling language and the creation of the proper implementation in the programming language.

A model is one of the key concepts used in natural and engineering sciences. In science, a model is meant to be a set of general assumptions, concepts and relationships that gives you a simplified way to describe a selected aspect of reality. A model is also a representation of the surrounding world in the mind of man, which, however, should not be confused with reality.

Models are created mainly for two reasons: to understand the problem domain better and to allow the exchange of information while the interested persons solve the problem. Undoubtedly, a significant amount of work and time to be devoted to the creation of an appropriate and correct model turns out to be an excellent investment; however, only provided that the model is properly devised and responsibly used. The increasing scope and complexity of problems faced by modern science and engineering more and more often call for a deliberate approach to solving them.

It turns out that in very many cases, a simple approach based on trying to constantly increase computing power, the number of people involved and funding devoted to solving the problem is far from enough. Modern science recognizes the need to use more effective and subtle ways. System modeling seems to be one of them. Since the correct description of the problem domain is always the basis for finding the solution, the key issue is its formulation that is exact and corresponds to reality. Modeling helps to identify the problem, to determine the scale of its complexity, to propose a solution and, which is especially important, to allow the flow of information between the interested parties in an excellent way. Modern science treats a model as a basis for communication between the parties interested in a given problem and involved in its solution.

If we wish to understand the problem domain (while describing a fragment of the existing reality), we should build a model of the problem domain. The purpose of this model is to create a correct abstraction of the real world. Such an abstract model should be as little complicated as possible, but it should still reflect the real world correctly, so that the behavior of entities in the real world could be predicted. The computer system is a collection of subsystems formed to perform a specific task and is described by some set of models, each of which describes a different aspect of reality.

The standard for the implementation of model-driven software MDD (Model-Driven Development) developed in the early 21st Century by the OMG organization (Object Management Group) is a very modern and technologically-advanced approach to the evolution of object-oriented software architecture. MDD uses model-driven development, which is an extension of the paradigm of model driven architecture, MDA (Model-Driven Architecture) in all aspects of the evolution of the system. MDA is not, strictly speaking, an entirely new methodology for software development. In fact, MDA is an extension of iterative development with the introduction of mechanisms for the automatic transformation of models. On the basis of the assumption of the standard for the implementation of model-driven software, the manufacturing process of the software for scientific purposes can be offered, consisting of the following stages:

Determination of the Problem Domain

(The process of creating concepts based on the general knowledge on the problem domain). This phase is intended to clarify the abstract concept that will become a real manufactured product in the future.

Development of a Mathematical Model of the Problem Domain

(The problem domain elements recorded in the language of mathematics).

Development of an Appropriate Numerical Algorithm

(In the proposed approach, the most important or all the essential algorithms for understanding the problem are recorded with the use of a formal UML notation in the form of appropriate activity diagrams).

High-level Modeling

In other words, it is a problem domain model (in terms of MDA, the equivalent would be a PIM platform-independent model). It is necessary to be aware of the fact that the domain objects are not the design objects. The problem domain model describes the way of functioning of the real world, and not how the computer system being created works. All models are made in the UML formalism.

Low-level Modeling

(In terms of MDA, the equivalent would be a model specific to the PIM platform). The previous phases have focussed on understanding the problem domain, while low-level modeling is focussed on creating an optimal solution. Low-level modeling is a process transforming the understanding of the requirements into a model that can be implemented in the form of software. The result of this process is to create a design document. A design document should be formally divided into two parts: class design and architecture mechanisms. Part of the class design is further divided into the statistical design (diagrams detailing the individual classes, their relationships, dependencies and characteristics) and dynamic design (diagrams describing how the objects of the individual classes work together, e.g. sequence diagrams).

***Corresponding author:** Andrzej Daniluk, Assistant Professor, Department of Applied Computer Science, Institute of Computer Science, Maria Curie-Skłodowska University, pl. M. Curie-Skłodowskiej 1, Lublin 20-031, Poland, E-mail: adaniluk@tytan.umcs.lublin.pl

Received August 10, 2012; **Accepted** August 13, 2012; **Published** August 16, 2012

Citation: Daniluk A (2012) Modeling in the Process of Constructing the Software for Scientific Purposes. J Inform Tech Softw Eng 2:e105. doi:10.4172/2165-7866.1000e105

Copyright: © 2012 Daniluk A. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

The design of architecture mechanism provides details of system-artifact implementation.

Coding

By using the appropriate MDA/MDD tools, the system model can be transformed into a code/framework that programmers can then complement using their skills/experience. It is also possible to write your own transformation rules using QVT standards.

Testing and Implementation

The suggested process may be iterative, which means the verification and validation possibility of the respective system components at each stage. It is possible to perform by means, e.g., of the relevant tools provided by the technologically-advanced MDA/MDD environments.

The use of this cycle in the process of constructing the software for scientific purposes results in:

· Shifting the burden of system development to a higher level of abstraction and giving a central role to modeling,

· Clear separation of the system layers,

· Automatic generation of a code directly from the model,

· Implementation of mechanisms for the automatic verification and validation of a code.