Engineers Discovering the “Real World” – From Model-Driven to Ontology-Based Software Engineering

Wolfgang Hesse
Fachbereich Mathematik und Informatik, Univ. Marburg, Hans Meerwein-Str., D-35032 Marburg
hesse@informatik.uni-marburg.de

Abstract. Going along with the successful dissemination of the Unified Modelling Language (UML) and the growing popularity of the Model driven Development (MDD) initiative models are occupying the world of Software Engineering. As prescriptive means they are not only changing – and maybe soon dominating - our approaches to software construction but in their descriptive function they have also considerably influenced our views on software application domains. Since software systems and their application domains are almost ubiquitous we can say: models reflect and represent our view on the real world. In philosophical terms, models have no longer just a technical end and purpose but they reflect our Weltanschauung, i.e. are part and centre of modern ontology.

In this contribution I shall focus on the ontological dimension of software modelling. Models are like windows between the outer, “real” world and our inner world of understanding – described and reflected by software. Software development is reality construction and with our application models we reconstruct our surrounding world. By this re-construction it is converted (and reduced) to a world of objects modelled as data capsules and maybe soon controlled by RFID chips. So called “ontologies” are the dictionaries of this construction process.

1 Introduction: The Prescriptive and Descriptive Role of Models

There is no doubt that models have become the central element of software development. The successful dissemination of the Unified Modelling Language (UML) and the growing popularity of the Model Driven Architecture/Development (MDA/MDD) initiative [M-M 03] are clear indicators of this methodological shift. Models have overtaken the role of specifications: Where twenty years ago methodologists debated about algebraic, operational, formal or semi-formal specifications now UML and MDA/MDD are the undisputed leaders in the software methodology scene.

Is this all just a renaming – old wine in new skins? Apparently not since models are at the same time more and less powerful than specifications. Less powerful since a specification is expected to be unambiguous and complete: it has to serve as a counterpart for the verification of any computer program built to fulfil the specification. It is an exact pre-image of what is to be built and an unmistakable measure of the correctness of the result. Models do often miss such strict quality criteria. On the other
hand, models can be more powerful than specifications insofar as they do not only appear as pre-images but also as after-images and, in particular, and play both roles at the same time.

Stachowiak has emphasised the relationship of each model to another object - its original - to which it is in some respect similar or equal (cf. [Sta 73, Lud 02]). Now it is important whether the model comes into existence before or after the original. In the first case we call it prescriptive and a pre-image, otherwise descriptive and an after-image. We can use a simple and intuitive notation (a modified form of that in [Küh 06]) to denote both situations:

(1.1) \( O \blacktriangleright \mathcal{M} \) ("\( O \) is modelled by \( \mathcal{M} \)) and \( \mathcal{M} \blacktriangleleft O \) ("\( \mathcal{M} \) is concretised to \( O \))

Examples of prescriptive models are the paper model house or a 3D visualisation produced by an architect or the above mentioned specification of a computer program. Examples of descriptive models are geographical maps or toy puppets, buildings or trains.

In modern Software Engineering, models often play a double role: They are pre- and descriptive at the same time. A UML model of a certain application domain or Universe of Discourse (UoD) - say a library - may represent this domain including books, journals, authors, borrowers and their relationships and it can serve as some sort of specification for the library administration software system to be developed. In general terms, a model \( \mathcal{M} \) stands for an original domain \( \mathcal{O}D \) and is at the same time a blueprint for a “system original” \( SO \) to be built:

(1.2) \( \mathcal{O}D \blacktriangleright \mathcal{M} \blacktriangleleft SO \)

In a previous paper I have compared this kind of models with Janus – the double-faced Roman God who guarded the entrance of almost every Roman patrician house (cf. [Hes 06]). Like Janus, UML models look at the same time outside (towards the application domain) and inside (to the software to be developed).

![Fig. 1. The double-faced God Janus on a Roman coin](image)

Model-driven Development and Model-driven Architecture (MDD/MDA, cf. [M-M 03]) are relatively new keywords which emphasize the prescriptive aspect of models. In MDA terms, a Platform-independent model (PIM) is taken as pre-image for building various