From “Formal Methods” to System Modeling

Manfred Broy

Institut für Informatik, Technische Universität München
D-80290 München Germany
broy@in.tum.de
http://wwwbroy.informatik.tu-muenchen.de

Abstract. When engineering software intensive systems the quality of the resulting product depends strictly on the quality of the models used explicitly or implicitly in the engineering process. A rich family of such models has been developed in recent years. We discuss some of these models and describe the requirements for system modeling theories.

1 Introduction

There is a long way from early approaches to formal system modeling sometimes called “formal methods” such as denotational semantics, VDM, SADT, algebraic specification to model based system and software engineering as advocated in widespread approaches such as UML, SysML, or MDA. We discuss this development and properties of the theory and methodology that we require.

System and software development is today one of the most complex and powerful tasks in engineering. Modern software systems typically are embedded in technical or organizational processes, distributed, dynamic, and accessed concurrently by a variety of independent user interfaces. Just by formulating the right programs we obtain engineering artifacts that can calculate results, communicate messages, control systems, and illustrate and animate all kinds of information. Since programs are implicitly or explicitly - based on models of system behavior and since well-chosen models are a successful way to understand software, modeling is an essential and crucial issue in software construction.

In all scientific and engineering disciplines, models play a prominent role. For physics, mathematics has provided lots of models. The same holds for many engineering disciplines. Economy works with models; biology works more and more with models, chemistry works with models. Constructing, analyzing, and arguing in terms of models is at the heart of science.

In informatics modeling is even more crucial. Developing software is more or less nothing than developing the right models finally represented in the appropriate notation such that they can be executed effectively and efficiently on today’s computing devices.

Many different models are needed in the engineering of software intensive systems. To name a few of them:

- Domain models: describing properties of the application domain that are relevant for the system under development (physical, technical, organizational, fiscal, legislation rules and laws),
• System models: describing the logical and technical behavior and structure of the system and software under development,
• Quality models: describing quality aspects of the system and software under development or its development process (see [3]),
• Cost models: calculating the cost and required budget for a development project,
• Development process models: describing the structure of the development process (see [5]).

These models are used as the basis for the engineering of software intensive systems. In particular, for the planning of a development project and in requirements engineering such models are indispensable. The formality of these models is only one aspect. Of course, if tool support is requested for the application of these models, formalization is inevitable.

![Fig. 1. Three Dimensions of Software Development](image)

In the following we concentrate rather on the system modeling aspect. In the development of large complex software system it is simply impossible to provide one comprehensive model for the system in only one step. Rather we

• specify a system or subsystem first in terms of its interface,
• add stepwise details by refinement,
• give several views,
• decompose the hierarchically system into components,
• construct a sequence of models on different levels of abstraction.

Each step in these activities introduces models, refines them, or integrates them. Concentrating on the modeling issues we have to manage the following tasks:

• selection of the appropriate model concept for an aspect,
• identifying and documenting all the properties for a model,
• integrating several views into an overall model,
• decomposing a model hierarchically into components.