On the evolution of conceptual modeling

Roland Kaschek

School of Engineering and Advanced Technology

Massey University

Wellington, New Zealand

Abstract

Conceptual modeling is the activity of creating conceptual models, i.e., models that describe problem structures in a way that still is relatively independent from the technology and strategy used to solve the problem. In that sense conceptual modeling has a rather long tradition and is intimately connected to the genesis of mathematics. Models deliver their problem solution aid largely by providing a guiding metaphor. For example, the guiding metaphor of mathematical models has been that of equation. Newer such guiding metaphors that are closely related to the genesis of informatics are algorithm, relation, intelligent system, and dialog. The latter guiding metaphors of model construction have found their utilization in software engineering, data engineering, artificial intelligence, and web engineering.

The conceptual modeling movement, at the begin of the 1980s, started out as a response to increasing specialization of those involved in software engineering, data engineering, or artificial intelligence. Its purpose was overcoming the unnecessary parts of the emerging divide between these sub-disciplines of computing. Since that time the specialization has increased rather than decreased. The feeling, however, remains that differences found in modeling between the various sub-disciplines of computing are more related to notation and basic vocabulary rather than to the used concepts, the ways to utilize the models, or the ways the models are constructed. Similarly it occurs such that that considered over time in a given sub-discipline of computing the differences that one finds are more related to notation, terminology, and implementation technology rather than the key model ingredients of modeling procedures.

This paper suggests using the evolution metaphor for conceptual models because the latter, like biological organisms, are subject to many different impacts of players involved in the model life cycle and the question arises whether or not any kind of pattern can be identified in that life cycle or the structure of conceptual models.

Since the 1980s the need increased for overcoming idiosyncrasies of approaches to modeling in the various sub-disciplines of computing, as there constantly emerge new such disciplines and (at least in some parts of computing) the need for participatory and interdisciplinary approaches to software development becomes more pertinent. At the same time programming language technology matures so that high-level programming language are becoming available that lack any relationship to

Dagstuhl Seminar Proceedings 08181
The Evolution of Conceptual Modeling
http://drops.dagstuhl.de/opus/volltexte/2008/1596

implementation technology. A respective example is the Olivanova programming machine that creates applications out of UML specifications.

Approaches to the foundations of modeling in the past often have solely relied on model semantics as provide by some sort of mathematical encoding of model terms. That is counterproductive in the sense of wide spread acceptance of such approaches, as the majority of folks developing and using software do not have the education required for understanding such approach. While one might pity that this fact won't go away, as software continues to penetrate all branches of human life it becomes ever more questionable to even aim at providing a foundation of modeling that rests on heavy use of mathematics. The paper therefore argues that model semantics should be looked at in a different way. One such way can be inspired by language games. At least two different language games relevant for software development can be identified easily: First, a cooperative language game in which application experts use software documentation and software for solving business problems. In this game the model semantics is constituted by business best practices and organization conventions. Second, there is a refutative language game in which model semantics in fact is formal. Not only is the model semantics different in these games while the model as a token, a structured pattern, or text is the same. Also the rules of the game are different. In the cooperative game first priority is given to getting a given job done with the available resources. In the refutative game, however, first priority is given to getting a correct implementation of a given specification. These two games are intermediated by a third game, i.e., the coordination game. In this game project leaders make sure that appropriate specifications are worked out in a sensible amount of time and that these then are implemented appropriately.

The paper argues then that none of the mentioned three language games can be made obsolete and that therefore conceptual modeling for software development needs to be considered from the view of each of these language games.

In the evolution of conceptual modeling thus not only syntactical changes appeared according to which the once favored concept of equation has experienced a reduced rating. Rather the usage of models has to be considered from a number of perspectives that were not relevant in a time when mathematical models were created and used by the same person and when model use essentially was manually performed by the model creator.