Reusable Simulation Models: An MDE-approach to Spatio-temporal Modeling with Cellular Automata

Falko Theisselmann[1,2] and Doris Dransch[2]
[1]Graduiertenkolleg METRIK, Department of Computer Science, Humboldt-Universität zu Berlin
[2]GeoForschungsZentrum Potsdam (GFZ), Telegrafenberg, Potsdam

INTRODUCTION

Numerous modeling tools, frameworks, and environments support domain experts with the implementation of spatiotemporal simulation models. The implemented models are usually bound to specific frameworks, because specific modeling languages, simulation engines, or processing platforms have to be used[1]. This reduces the degree of model reuse and integration, particularly in a multi-disciplinary context. Our approach addresses two issues: model implementation and model reuse. This approach is conceptually based on the suggestion of Evert et al., (2005) to separate the modeling level, on which simulation models are described by modelers, and the implementation level, on which models are linked and executed. The approach is realized using Model-driven Engineering (MDE) and is presented for modeling cellular automata (CA).

MODEL DRIVEN ENGINEERING FOR SPATIOTEMPORAL MODELING: A THREE-LEVEL APPROACH

In MDE, software is mainly described by models, not by code. Executable code is automatically synthesized from these models (Schmidt, 2006). In our approach, simulation software is explicitly modeled on three distinct levels of abstraction. At each level, the simulation software is represented by a level-specific model. The concepts that are available for modeling on the different levels are formally prescribed by level-specific metamodels that define the respective modeling languages.

On the highest, most abstract, level, a *domain specific model* is defined by the means of a domain specific modeling language (DSL). This DSL can be tailored to the needs of the modeler and should provide modeling concepts with a clear relation to the problem domain (i.e. hydrological modeling, fire spread modeling, land use change modeling). A domain specific model is automatically transformed to a *formalism specific model*.

The formalism specific model is modeled using the concepts of a modeling formalism that is more expressive in comparison to the DSL. Moreover, it holds more implementation and computation detail. For modeling on this level, we developed the hybrid cellular automaton (HCA) formalism as a generic modeling language for CA simulation models.

[1] In the remainder of this paper the term *framework* subsumes software that is named library, tool, framework, language and environment in literature.

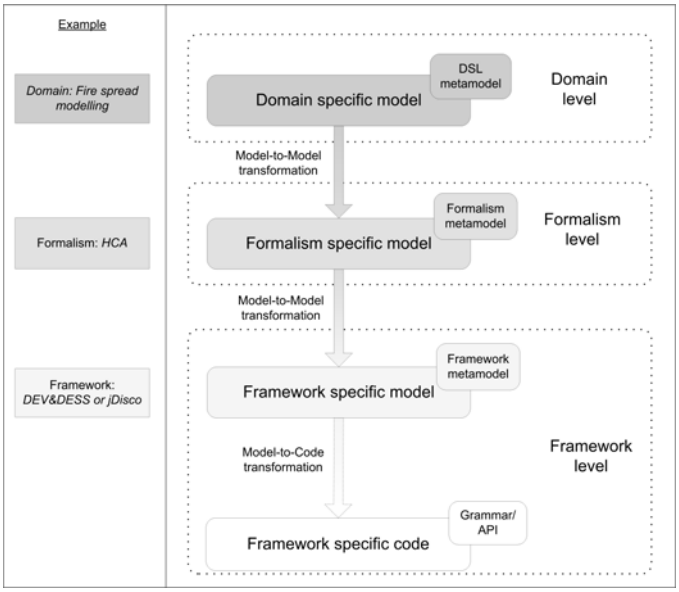

Figure 1: A MDE based three-level implementation approach to simulation model implementation.

A formalism specific model is automatically transformed to a *framework specific model* from which executable code is automatically generated. The concepts of the modeling language at the framework level are provided by a simulation framework (API, grammar). In this implementation approach, the modeling formalism on the framework level may provide greatest expressivity and holds most implementation detail.

Generally, with this approach model reuse can be realized by defining different transformations from the formalism level to different frameworks and the respective code generation. By this a model can be reused on different frameworks. The domain specific model, the formalism specific model, and the transformation between domain level and formalism level remain unchanged.

The approach is based on the assumption, that there is a common understanding on all modeling levels of how a system is to be modeled. This common understanding is reflected in the basic modeling concepts. In general, we adopt the notion systems theory and event-based modeling. Models may be constructed by assembling interacting submodels, where each model has a state and may exhibit discrete and continuous behavior. A model's state and discrete state transitions can be defined along the lines of general purpose programming languages.

Using HCA, a CA is composed of interacting cell-submodels. Figure 2 illustrates a possible trajectory of a single cell, where the model's state evolves continuously between events. At times of events (*t1*, *t2*) the model's state changes instantaneously.

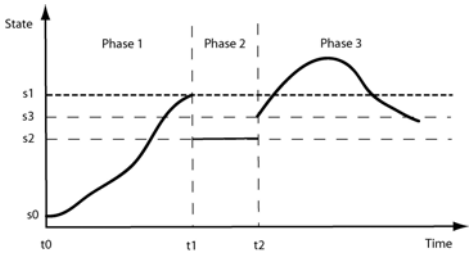

Figure 2: The hybrid behavior of cells of the HCA.

Based on the basic modeling concepts, CA specific extensions allow for the specification specific elements, such as the definition of a cell, a neighborhood, and transition functions based on a neighborhood (Figure 3).

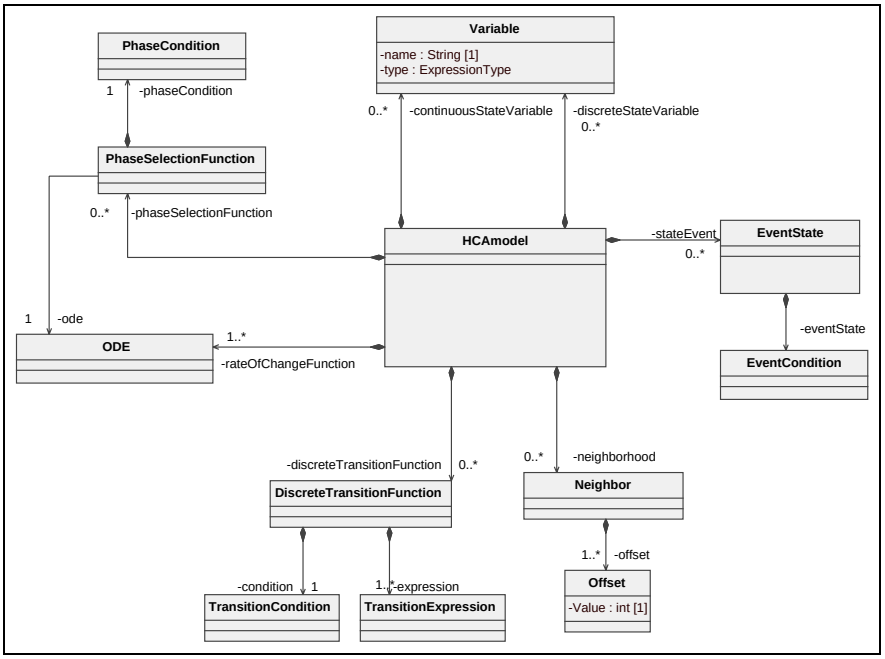

Figure 3: A partial metamodel for the HCA formalism on the formalism level (UML notation).

For model execution, we use the process-based simulation framework jDisco (Helsgaun, 2001) that provides means to realize the basic and CA specific modeling concepts. Since the required execution functionality is generic, model execution may also be realized using other generic simulation frameworks i.e. DEV&DESS (Praehofer et al, 1993). For this, specific metamodels and transformations had to be defined.

CONCLUDING REMARKS AND OUTLOOK

We developed MDE approach to modeling CA that is based on generic simulation functionality. The core element of this approach is the HCA - a generic formalism to model CA. Currently, this

formalism allows the definition of simple models, like Conway's Game of Life. More expressivity will be added by extending the means to define transitions and to query the CA's state (i.e. statistical functions). Moreover, model coupling is to be introduced to integrate CA with other models (i.e. workflows, agents). Currently, the HCA can directly be used for modeling. However, as the HCA's expressivity is extended, it will become harder to use. Because of this, possibly less expressive but more intuitive languages should be provided on the domain level for particular classes of CA.

Prototypical implementations show that model performance is a key challenge, since model execution is demanding and efficient simulation is crucial to make this approach feasible for big cellular automata. Moreover, the inclusion of GIS functionality is needed. Today's GIS standards and generic GIS technologies may provide the basis for the integration of GIS functionality with simulation modeling using an MDE-approach.

ACKNOWLEDGEMENTS

The presented work is supported by Deutsche Forschungsgemeinschaft, Graduiertenkolleg METRIK (GRK 1324/1).

BIBLIOGRAPHY

Evert, F. van, Holzworth, D., Muetzelfeldt, R., Rizzoli, A., and Villa, F. (2005) Convergence in integrated modelling frameworks. In Zerger, A. and Argent, R.M. (eds) MODSIM 2005 International Congress on Modelling and Simulation. Modelling and Simulation Society of Australia and New Zealand, pp. 745-750.

Helsgaun, K. (2001) jDisco - a java package for combined discrete event and continous simulation. Technical report, Department of Computer Science, Roskilde University.

Praehofer, H. Auering, F. and Reisinger G. (1993) An Environment for DEVS-Based Multi-Formalism Simulation in Common Lisp/CLOS, Discrete Event Dynamic Systems: Theory and Applications, Volume 3, 1993, Pages 119-149.

Schmidt D. (2006) Model-driven Engineering. Computer, 2/2006, Pages 25-31.