Computational Methods for Modeling Migration

Oliver Reinhardt and Adelinde M. Uhrmacher
oliver.reinhardt@uni-rostock.de, adelinde.uhrmacher@uni-rostock.de
Institute of Computer Science
University of Rostock

Abstract

Developing simulation models is an intricate process, particularly, if the goal is to develop a comprehensive simulation model of international migration. Therefore, suitable computational support is needed. This may involve using domain-specific modeling languages that allow a compact model description and a composite model design, and methods to make a model's foundations in data sources and theories, and data generating processes explicit. In this paper, we compare the encoding of a routes-rumors model of migration in ML3, a domain-specific modeling language for demography with a focus on modeling linked lives in continuous time, and the encoding in the general purpose language Julia. Additionally, we present a provenance model for the routes-rumors model, which relates the diverse artifacts and processes involved in generating the model, including the diverse simulation experiments. The examples elucidate the potential and requirements of the methodological approach for facilitating the generation, maintenance, and reuse of simulation models.

1 Introduction

Migration is one of the most uncertain demographic processes. Migration processes are not only influenced by changes of the physical, social, and political environment, which alone are already difficult to predict, but also by a sequence of individual decisions that depend on the individual, and their own experiences. This inherent complexity must be reflected in any sufficiently accurate model of human migration [23], which provides a formidable challenge for any modeling and simulation of migration processes.

Therefore, no single model will suffice. Instead, multiple models which focus on different aspects of migration need to be developed and combined. These models have to be be based on various theoretical foundations, sources of data, and, last but not least, psychological experiments. Simulation experiments for calibration, validation, sensitivity and uncertainty analysis will help refining the simulation models, which will undergo various changes within this process. Due to the complexity of the model building processes, suitable computational support is required.

In this work we will focus on a) the role of declarative domain-specific modeling languages to support a compact description and a composite design of simulation models and b) explicit means to clarify the relation between simulation models and other artifacts that have been used during developing a simulation model.

To illuminate our methodological approach, we will use the first model developed in the project, the routes-rumors model (details provided in a separate paper on model design specification), which is concerned with how migration routes form. Thereby, it is of interest, how migrants get, perceive and share information about the resources and dangers of the landscape they move through, and how they decide on their movement given this information.

2 Domain Specific Modeling Languages

Designing and implementing a complex simulation model from scratch is a difficult task, requiring deep knowledge of the modeled system as well as software development skills. Domain specific modeling languages (DSMLs) are a means to ease this task. A domain specific language is a programming language that is not designed as a general-purpose tool, but to solve specific problems of a specific application domain [26]. A DSML is designed to implement simulation models in a specific domain. It makes use of domain metaphors (e.g., agents or social networks), focuses on supporting certain model properties (e.g., grid space or continuous space), and contains build-in solutions for typical problems of the domain (e.g., aging and age-dependent behavior).

DSMLs have been developed for and successfully applied in a wide variety of application domains, e.g., in biochemistry [7] or digital systems [11]. For agent-based models in discrete time and grid space, NetLogo [22] is commonly used. The Modeling Language for Linked Lives (ML3, [21]), was developed for continuous-time agent-based models with dynamic social networks in the social sciences, especially in demography. Many of these languages, e.g., [1, 8], have a formal semantics, mapping the model onto some kind of mathematical structure, e.g., a Continuous-Time Markov Chain.

To demonstrate the approach, Figures 1 and 2 show a snippet of the routes-rumors model implemented in Julia, a general purpose programming language, and the aforementioned DSML ML3. Both snippets show the movement of a migrant through the landscape: The migrant decides for a target in the environment of their current location (this decision process is not shown, but happens inside the function called in line 4 (Figure 1) and line 3 (Figure 2)). They move to that location and the cost of that movement is deducted from their capital. Apparently, the ML3 implementation is much more concise. This can mostly be attributed to two reasons: Firstly, all function calls in the Julia code have a list of parameters including the model state (`world`), the migrant (`agent`), and the model parameters (`par`). As those are common concepts to all simulation models, in ML3, they need not be handled explicitly. Secondly, the Julia snippet contains the actual movement logic (line 1–9, line 13) and a corresponding bit

```
1  Migrant
2   | ego.capital >= 0
3   @ ego.move_rate()
4   -> ?target := ego.decide_move_target(),
5      ego.capital -= Migrant.move_cost(?target),
6      ego.location := ?target;
```

Figure 1: An excerpt of the ML3 implementation of the routes-rumors model showing the migrant's movement.

```
1  # model logic:
2  function step_agent_move!(agent, world, par)
3        loc = decide_move(agent, world, par)
4        if loc == Pos(0, 0)
5              return
6        end
7        costs_move!(agent, find_location(world, loc.x, loc.y), par)
8        move!(world, agent, loc.x, loc.y)
9  end
10
11 # corresponding scheduling logic:
12 function step_agent!(agent :: Agent, model::Model, par)
13       if agent.capital < 0.0 || decide_stay(agent, par) # model logic
14             step_agent_stay!(agent, model.world, par)
15       else
16             step_agent_move!(agent, model.world, par)
17       end
18       step_agent_info!(agent, model, par)
19 end
```

Figure 2: An excerpt of the Julia implementation of the routes-rumors model showing the migrant's movement.

of scheduling logic (lines 11–19) that makes sure that the movement logic is executed at the appropriate time. These two parts of logic are tightly interwoven. In ML3 the scheduling logic need not be implemented by the modeler, it is part of the simulator.

This separation of the model from the simulator, see [25], is one of the central advantages of using a DSML. Using a general purpose language, the modeler has to implement the model logic, e.g., the decision processes of agents, and in addition the execution logic, e.g., scheduling events or advancing time. Using a DSML, the model logic remains the responsibility of the modeler, while the simulation logic is implemented by the developer of the modeling language. In the ML3 example in Figure 1, the modeler only specifies which agents are able to move (line 2) and with what rate they move (line 3), corresponding to the two conditions in line 13 of the Julia snippet, but the actual scheduling is handled transparently to the modeler. A clear separation between model and simulator makes the model implementation more succinct, and therefore more accessible. As there is less implementation effort for the modeler, there is also less room for errors. Additionally, as the simulation logic is now independent of the model, it can be reused for multiple models. This allows for putting more effort into implementing and maintaining the simulation logic, improving software quality and enabling the implementation of more efficient advanced simulation algorithms [9]. Thereby, the focus on a specific application domain allows for the simulation algorithm to exploit domain-specific properties of models for a more efficient execution. The formal semantics unambiguously specifies the semantics of a simulation model defined in this language and also serves as a blueprint for designing the simulator.

More complex simulation models are typically composed of simpler models [15]. In a model implemented from scratch in a general programming language, a composite model design is more difficult to realize, as model logic as well as execution logic has to be considered. In ML3, a model is composed of different agents of various kinds, which encourages composite model design at agent level. Additionally, each agent's behavior is the result of composing a set of independent stochastic rules, each governing a specific part of the agent's behavior, enabling composite design at the level of behavioral patterns. Model components can be developed, calibrated, and validated separately, to be later composed to the complete model [16, 17]. A composite design of simulation models also facilitates exchanging model parts, for example, to compare different decision-making strategies in a multi-model approach as proposed in [4].

All together, these properties of DSMLs ease development and implementation of the model, its later extension, and, due to improved accessibility, its dissemination. With an external DSML like ML3, where the DSML is a completely independent language, this comes at a cost of reduced flexibility. Model properties and features not accounted for by the DSML might be difficult or impossible to realize. Internal DSMLs, e.g., [10], can offer a middle ground between external DSMLs and general purpose languages. Here, the DSML is embedded in a general purpose host language. Models implemented in an internal DSML are host language programs. Consequently, host language development tools can be used, and the modeler might fall back to using host language features if the DSML is too restrictive. As a trade-off, the syntax of an internal DSML must be compatible with the host language, often leading to less concise and accessible model code. Also, the advantages of using a DSML will be partially lost if the modeler has to make extensive direct use of the host language.

3 Managing Provenance

To understand and to trust a simulation model, ample information, e.g., about the model, its foundations, and its validation, is crucial. Hence, the simulation community has developed several standards to manage it. Most commonly applied in agent-based modeling is the ODD protocol (Overview, Design concepts, Details; [5]). The protocol prescribes certain content and structure for a textual description of the model, e.g, containing the model's purpose, the model entities and processes, model components, and input data. However, ODD has significant shortcomings. An ODD document only describes a single version of a single model, having an iterative modeling process or having multiple interdependent models is not supported by the standard. Also, the documentation of simulation experiments is not addressed. For the modeler, writing and maintaining an ODD document, which amounts for dozens of pages of text for a complex model, is a significant burden. Finally, a verbal description is often imprecise, and, as it has to be maintained manually, error-prone.

Provenance models offer a more structured representation of this information – provenance being defined as the "information about entities, activities, and people involved in producing a piece of data or thing, which can be used to form assessments about its quality, reliability, or trustworthiness"[6]. Using the Open Provenance Model standard (OPM, [14]), provenance information is described as a directed graph. Nodes represent *artifacts*, i.e., digital representations of entities, e.g., models, model components, or data sets, and *processes*, i.e., activities performed with artifacts, generating new artifacts, e.g., extending a model, or conducting an experiment. Edges represent relations between the nodes, and edges between different types of nodes represent different kinds of relations, most importantly:

- an artifact is *used by* a process, e.g., a simulation model is used, when a simulation experiment is performed
- an artifact is *generated by* a process, e.g., when a simulation experiment is performed, some output data is produced

Managing the provenance of data has been researched for at least two decades, and a variety of software tools and platforms have been developed, e.g., Fairdom [24] and VisTrails [2]. However, in these approaches, the simulation model is only viewed as part of the provenance of the simulation data. The provenance of the simulation model itself, i.e., the process of creating the simulation model, has only recently been put into the focus [19, 20].

Figure 3 shows a mock-up example, of how a provenance model for the routes-rumors model could appear. In a complete provenance model, all the artifacts and processes would be annotated with information about them, and the relations would be annotated with the roles the nodes have in the relation. The initial version of the routes-rumors-model is depicted as the artifact RR_0. It is produced by composing (process *comp.*) several model components, e.g., the landscape the migrants move through C_{space}, the information exchange C_{info}, or the migrant's decision-making C_{dec}. Please note, that this is a rather high level view of the modeling process, focusing on the different model components and their foundations, but ignoring the evolution of the model and its components over time. Depending on the questions the provenance model shall answer, a more abstract or more detailed view might be advantageous. The provenance of the decision component is depicted in more detail. It is created by parameterizing (*par.*) a preliminary unparameterized version M_{dec}, which itself is created by some (again rather abstract) modeling process (*mod.*) based on Prospect Theory (PT) [12], as the psychological theory of decision-making at its heart, and other foundations.

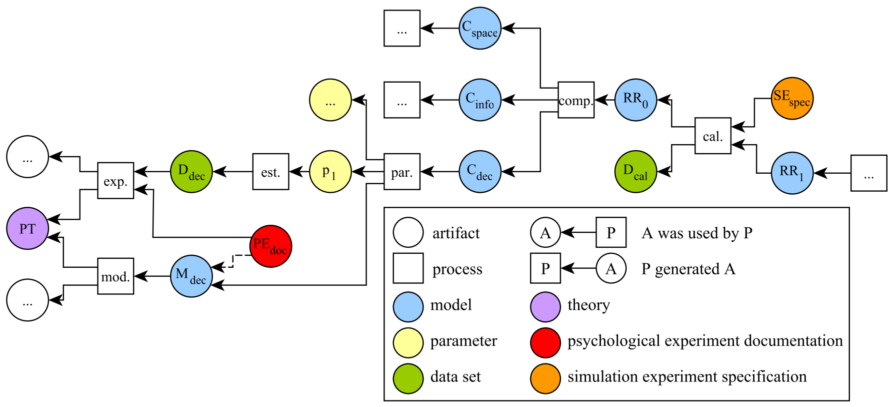

Figure 3: A mock-up provenance model for the routes-rumors model. The figure only shows an excerpt of a potential provenance model, and omissions are marked with "...". All the artifacts and processes are explained shortly in the text.

Furthermore, the provenance model shows the origin of one of the parameter values (or one set of parameter values) p_1. It is estimated (*est.*) from a data set D_{dec}, which was produced in a psychological experiment. This experiment is shown as a process (*exp.*), which is based on Prospect Theory (PT) and some other foundations, and produces the data as well as a documentation of the experiment (PE_{doc}). Thereby, we have abstracted from the experimentation process, i.e., designing, preparing and performing the experiment, and put an emphasis on its most important outputs: the data and the documentation, which allows to assess the data and reproduce the experiment.

Finally, a provenance model can include simulation experiments, which, for example used for calibration and validation, form an integral part of a simulation model's provenance. In Figure 3, a possible calibration experiment conducted with the routes-rumors model is shown on the right side. The experimentation is shown as a process (*cal.*), which uses the initial version of the routes-rumors model RR_0 and some input data D_{cal} as calibration target, and produces a calibrated version of the model RR_1 and a specification of the conducted simulation experiment SE_{cal}. Here, experiment specification means any set of instructions that allows to repeat the steps of the experiments and reproduce the results.

While a provenance model can be used to give a graphical overview about a simulation model's structure and origin, as has happened here, its true purpose is a more systematic computational handling of that information. Inference mechanisms such as the query language OPQL [13] allow for answering questions about the model's provenance, e.g.: Which artifacts depend on a certain data set in which we have detected an error? What data was used for calibrating the model? Or, what theoretic foundation justifies a certain model mechanism?

Also, common patterns in provenance models can be exploited. For example, the pattern on the right side of Figure 3 (simulation experimentation process producing an experiment specification) can be exploited to automatically generate a package containing all necessary information to reproduce the simulation experiment. Thereby, the experiment specification and all artifacts used by the experimentation process have to be included. Executing the steps in the experiment specification should then reproduce the result artifact, here the calibrated model. If the experiment specification is executable, e.g., using a domain-specific language for simulation experiment specification [3], the process of reproducing the results can even be automated. Examples of such executable packages of simulation experiments using a model of migration and a model of social care can be found in [18].

4 Open Questions and Next Steps

With domain specific modeling languages and provenance models, we have proposed two computational methods to ease managing the intricate process of modeling migration. However, open questions remain, both regarding basic methodology and the application to migration research.

As DSMLs are always designed for a specific kind of simulation models, the choice of DSML has to be carefully considered. An unsuitable DSML that does not account for central properties of the simulation model does not help but hinder model development. For migration modeling, important criteria might include spatial resolution, e.g., grid-based, graph-based or continuous space, temporal resolution, e.g., discrete or continuous time, or the complexity of the agents' learning and decision-making processes. In the next stages of the project, we will ultimately compare the Julia-implementation of the routes-rumors model with an equivalent implementation in ML3, which we chose as a candidate for a DSML. This will guide us in finding the right DSML to use, to expand upon, or, if no such language exists, to develop.

While provenance models can be constructed manually, as has happened in this example, the formal structure of provenance models allows for automatic or semi-automatic capturing and management of provenance information. To that end, an ontology of modeling and simulation provenance is necessary. As a first step, central artifacts as well as the processes and their relations need to be identified. In [19, 20], a set of artifacts and processes relevant for simulation studies has already been identified (see also Figure 3). Sources to collect provenance information could then be found in annotations to the simulation model, psychological experiments, and data, in scripts that execute the simulation experiments, in version control systems, or by monitoring the modeler's workflow. Many of these sources will provide provenance information on a relatively fine level of granularity, resulting in very large provenance graphs. To handle these provenance graphs effectively, techniques for aggregating on demand will become necessary.

References

[1] L. Bortolussi, R. De Nicola, V. Galpin, S. Gilmore, J. Hillston, D. Latella, M. Loreti, and M. Massink. “CARMA: Collective Adaptive Resource-Sharing Markovian Agents”. In: *Electronic Proceedings in Theoretical Computer Science* 194 (2015), pp. 16–31.

[2] S. P. Callahan, J. Freire, E. Santos, C. E. Scheidegger, C. T. Silva, and H. T. Vo. “VisTrails: Visualization Meets Data Management”. In: *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’06. New York, NY, USA: ACM, 2006, pp. 745–747.

[3] R. Ewald and A. M. Uhrmacher. “SESSL: A Domain-Specific Language for Simulation Experiments”. In: *ACM Transactions on Modeling and Computer Simulation* 24.2 (2014).

[4] J. Gray, J. Hilton, and J. Bijak. “Choosing the Choice: Reflections on Modelling Decisions and Behaviour in Demographic Agent-Based Models”. In: *Population Studies* 71.sup1 (2017), pp. 85–97.

[5] V. Grimm, U. Berger, D. L. DeAngelis, J. G. Polhill, J. Giske, and S. F. Railsback. “The ODD Protocol: A Review and First Update”. In: *Ecological Modelling* 221.23 (2010), pp. 2760–2768.

[6] P. Groth and L. Moreau. *PROV-Overview – An Overview of the PROV Family of Documents*. Technical Report. World Wide Web Consortium, 2013.

[7] L. A. Harris, J. S. Hogg, J.-J. Tapia, J. A. P. Sekar, S. Gupta, I. Korsunsky, A. Arora, D. Barua, R. P. Sheehan, and J. R. Faeder. “BioNetGen 2.2: Advances in Rule-Based Modeling”. In: *Bioinformatics* 32.21 (2016), pp. 3366–3368.

[8] T. Helms, T. Warnke, C. Maus, and A. M. Uhrmacher. “Semantics and Efficient Simulation Algorithms of an Expressive Multilevel Modeling Language”. In: *ACM Transactions on Modeling and Computer Simulation* 27.2 (2017).

[9] J. Himmelspach and A. M. Uhrmacher. “Plug’n Simulate”. In: *Proceedings of the 40th Annual Simulation Symposium*. IEEE, 2007, pp. 137–143.

[10] R. Honorato-Zimmer, A. J. Millar, G. D. Plotkin, and A. Zardilis. “Chromar, a Language of Parameterised Agents”. In: *Theoretical Computer Science* (2017).

[11] IEEE. “VHDL Language Reference Manual”. In: *IEEE Standard 1076-2008* (2009).

[12] D. Kahneman and A. Tversky. “Prospect Theory: An Analysis of Decision under Risk”. In: *Econometrica* 47.2 (1979), pp. 263–291.

[13] C. Lim, S. Lu, A. Chebotko, and F. Fotouhi. “OPQL: A First OPM-Level Query Language for Scientific Workflow Provenance”. In: *Proceedings of the 2011 IEEE International Conference on Services Computing*. IEEE Press, 2011, pp. 136–143.

[14] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. V. den Bussche. “The Open Provenance Model Core Specification (v1.1)”. In: *Future Generation Computer Systems* 27.6 (2011), pp. 743–756.

[15] E. H. Page and J. M. Opper. “Observations on the Complexity of Composable Simulation”. In: *Proceedings of the 1999 Winter Simulation Conference*. IEEE, 1999, pp. 553–560.

[16] D. Peng, T. Warnke, F. Haack, and A. M. Uhrmacher. "Reusing Simulation Experiment Specifications in Developing Models by Successive Composition — a Case Study of the Wnt/β-Catenin Signaling Pathway". In: *SIMULATION* 93.8 (2017), pp. 659–677.

[17] M. E. Pierce, U. Krumme, and A. M. Uhrmacher. "Building Simulation Models of Complex Ecological Systems by Successive Composition and Reusing Simulation Experiments". In: *Proceedings of the 2018 Winter Simulation Conference*. IEEE, 2018.

[18] O. Reinhardt, J. D. Hilton, T. Warnke, J. Bijak, and A. M. Uhrmacher. "Streamlining Simulation Experiments with Agent-Based Models in Demography". In: *Journal of Artificial Societies and Social Simulation* 21.3 (2018).

[19] O. Reinhardt, A. Ruscheinski, and A. M. Uhrmacher. "ODD+P: Complementing the ODD Protocol With Provenance Information". In: *Proceedings of the 2018 Winter Simulation Conference*. IEEE, 2018.

[20] A. Ruscheinski and A. Uhrmacher. "Provenance in Modeling and Simulation Studies – Bridging Gaps". In: *Proceedings of the 2017 Winter Simulation Conference*. IEEE, 2017, pp. 872–883.

[21] T. Warnke, O. Reinhardt, A. Klabunde, F. Willekens, and A. M. Uhrmacher. "Modelling and Simulating Decision Processes of Linked Lives: An Approach Based on Concurrent Processes and Stochastic Race". In: *Population Studies* 71.sup1 (2017), pp. 69–83.

[22] U. Wilensky. *NetLogo*. https://ccl.northwestern.edu/netlogo/. 1999.

[23] F. Willekens, D. Massey, J. Raymer, and C. Beauchemin. "International Migration under the Microscope". In: *Science* 352.6288 (2016), pp. 897–899.

[24] K. Wolstencroft, O. Krebs, J. L. Snoep, N. J. Stanford, F. Bacall, M. Golebiewski, R. Kuzyakiv, Q. Nguyen, S. Owen, S. Soiland-Reyes, J. Straszewski, D. D. van Niekerk, A. R. Williams, L. Malmström, B. Rinn, W. Müller, and C. Goble. "FAIRDOMHub: A Repository and Collaboration Environment for Sharing Systems Biology Research". In: *Nucleic Acids Research* 45.D1 (2017), pp. D404–D407.

[25] B. P. Zeigler, H. Praehofer, and T. G. Kim. *Theory of Modeling and Simulation*. 2nd. San Diego, CA, USA: Academic Press, 2000.

[26] A. van Deursen, P. Klint, and J. Visser. "Domain-Specific Languages: An Annotated Bibliography". In: *SIGPLAN Notices* 35.6 (2000), pp. 26–36.