Validation of Multi-Level Simulation Models in Demography – Making it Explicit

Oliver Reinhardt, Tom Warnke, Andreas Ruscheinski, and Adelinde M. Uhrmacher

University of Rostock, Institute of Computer Science, Albert-Einstein-Straße 22, 18059 Rostock, Germany
{oliver.reinhardt,tom.warnke,andreas.ruscheinski,adelinde.uhrmacher}@uni-rostock.de

Abstract. The validation of multi-level models in demography or sociology is a challenging task. Typically, to develop such a model, we speculate at micro level and calibrate at macro level. If possible, the micro level is rooted in established theories and some data. To increase the trust into the model, diverse simulation experiments are executed to explore the behavior of the model and to check its plausibility. Thus, these simulation experiments present an important information about the validity of the model, similarly as the data used for calibration, as input for the model, and for testing its predictiveness. Multi-level models are rarely developed from scratch but by reusing existing models, e.g., by extending or composing them, or for cross-validation. These models and their validity provide further details about the validity of a multi-level model. Thus, a multitude of artifacts contribute intricately related to the final multi-level model and our "gut feelings" about it.
To make these artifacts and their relations explicit and accessible, we will apply a declarative formal modeling language for simulation, a declarative language for specifying and executing diverse simulation experiments, and a provenance model to relate the diverse artifacts in telling the validation tale of an agent-based migration model.

Keywords: validation, multi-level modeling, demography, provenance

1 Introduction

Many natural systems are hierarchically organized, e.g., cell biological systems from proteins and cells up to organs, or social systems from individuals and social groups up to societies. Thereby, the lower level influences the upper level and vice versa. These up- and downward causations are central for multi-level systems and their modeling [37]. Multi-level models in the social sciences generally consist of micro-level entities, often individual persons, and higher level actors like households, or entire societies. Changes at population level depend on the action and interaction of individuals, social groups, and families, as they are embedded in their macro-level context. For example, the aggregate-level distribution of age at marriage can be explained through the process through which individuals decide whom and when to marry. However, the context, the common marriage age in a society, does not only form due to the decisions at micro level but also influences the decision of individuals. Thus, we find both upward and downward causation at work in demographic systems.

Based on the work of Colemann [13], Billari [8] identifies a two-stage process as essential for developing demographic models. In a first step, an unusual phenomenon is observed in the macro-level data. Often phenomena cannot be explained by only referring to the organizational level at which they have been observed: "Explanation of observed behaviour is not possible with reference solely to the spatial-temporal scale at which the observation was made" [54]. Therefore, as a second step, a hypothesis about the unobserved micro-level actions and interactions of individuals is formed, which shall replicate the macro-level observation. A wide range of demographic simulation models follows this pattern, e.g., for fertility prediction [17], migration [25,28], or partnership formation (e.g. the Wedding Ring [6,7], marriage markets [56]).

Validation is an important part of the modeling and simulation life cycle, as validation helps deciding whether a useful approximation of the system has been achieved, and directs a model's further refinement and enrichment, or as stated by Osman Balci: "Model *Validation* is substantiating that the model, within its domain of applicability, behaves with satisfactory accuracy consistent with the M&S objectives. Model validation deals with building the right model." [2, p. 135]. Clearly validation is an approximative process, with which the trust into a model is successively increased and, correspondingly, different approaches exist. Zeigler defines three levels of validity [55]: *replicative validity* or *historical validity*, i.e., the model reproduces data which has been observed from the real system (retrodiction), *predictive validity*, where a model produces data before it is observed from the real system, and *structural validity*, where the model reflects the structural relations of the real system. Troitzsch's discussion [49] about the question, "whether a theory which predicts empirical observations correctly at the same time explains what it predicts", deals with the difference between the second and third level of Zeigler's approach: explaining being interpreted as "showing how things work". The replicative validity is also called calibration of the model, and predictive validity refers to testing whether the model is able to reproduce data it has not seen, i.e., been trained with, before.

Data is obviously central for model validation, i.e., as model input, for calibration, and to test its predictive power. In any case, a thorough analysis of the model, its structure and its behavior, furthers the trust into the model. Therefore, a plethora of methods are available. Non-experimental approaches for inspecting a model's content and structure, e.g., by code walkthrough. Other methods focus on a model's dynamics, e.g., by trace inspections and analysis methods. Some of these methods rely on data to compare the model with, or on other simulation models whose dynamics the model should converge to or diverge from [48], some employ behavioral requirements specified as temporal logics [41], and some might analyze the sensitivity to changes in parameter values, screening the model's behavior [29]. Thus, a lot of different validation procedures, some non-experimental, some experimental, are applied to a model for validation. Many of these procedures rely on other artifacts. To assess the range of questions that a model is likely to provide valid answers for, the accessibility of this information is of vital importance, and, given the diversity of validation processes and artifacts, anything but trivial. In the following, we will explore how domain-specific languages for modeling, for specifying simulation experiments, and for keeping track of provenance allow us to approach this problem. We will demonstrate this based on retracing part of the development process of a concrete demographic model.

2 Migration - Complex Decision Processes

The model presented by Klabunde et al. [27,28] explores the hypothesis that in a critical phase approximately between the ages of 18 and 40, individuals make a series of important life decision, e.g. to get married or to have children, with which the decision to migrate competes. In simulations it is tested whether based on this micro-level hypothesis the observed age pattern of migrants (macro-level data) can be explained. Thereby, the linked life courses of individuals are in the focus. This includes marriage, fertility, and mortality of individuals, which are governed by stochastic rates, as well as income and expenses. The migration decision process itself is modeled based on the Theory of Planned Behavior [1]. The assumption is, that the decision to migrate is made in multiple stages, through which every potential migrant goes (see Figure 1): an intention is formed, plans and then preparations are made, and finally the migration is attempted. Each agent has an intention to migrate, which, in accordance with the Theory of Planned Behavior, is derived from their attitude towards migration, their beliefs about social norms regarding migration, and their beliefs about behavioral control regarding migration. Those three factors are influenced by the agent's personal situation and his or her environment. A total of eight free weighting parameters determines the strength with which different aspects influence the migration intention. Finally, the migration intention governs how fast the agent proceeds through the stages of the decision process, as shown in Figure 1.

The model was then applied to the case of migration from Senegal to Europe. To this end marriage, fertility, mortality, income, and expenses were estimated from data. For marriage a Coale-McNeill model [12] was fitted, using data from the Demographic and Health Survey of Senegal (DHS) for individuals in Senegal, and from the MAFE survey (Migration between Africa and Europe) [4] for individuals who migrated. The individuals are then paired by employing a marriage market [56]. Fertility was also estimated from DHS and MAFE data. For mortality a Heligman-Pollard model [24] was fitted to data from the UN World Population Prospects 2015. Income is taken from IMF data, consumption from World Development Indicators. An initial population was sampled from the 1988 Senegal census. Initial wealth was estimated from data by Davies et al. [15] By adjusting the 8 free parameters the model was then calibrated to reproduce the distribution of the age at migration and the distribution of the time passed

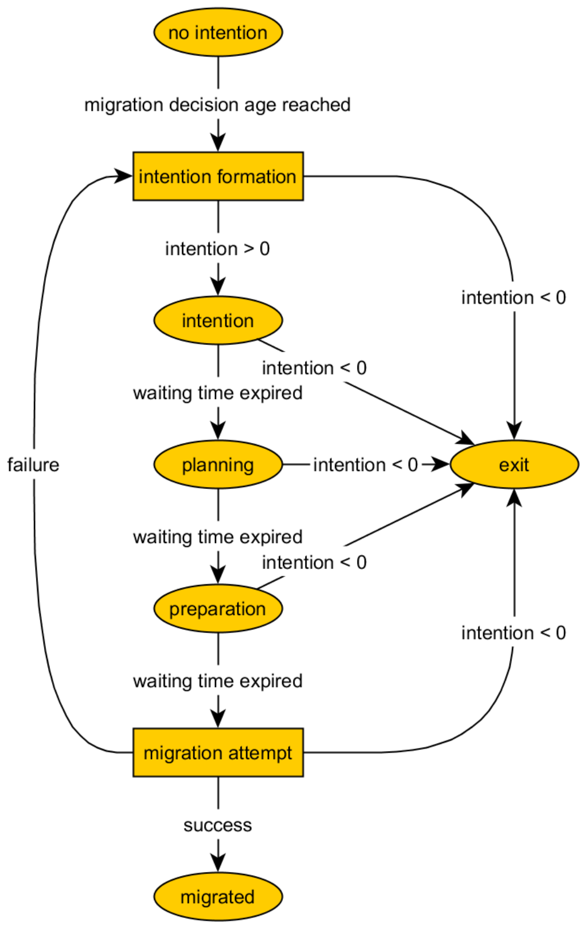

Fig. 1: The stages of the migration decision process. When a person is born, they start in the "no intention" stage. When they reach a certain age, they enter the intention formation process. As long as the migration intention is positive, they advance through the stages, until they attempt migration. The waiting times until hey advance are stochastic and depend on the strength of their intention. When their intention gets negative, they leave the decision process.

between starting to plan migration and the actual migration attempt observed in the MAFE survey. Furthermore, a sensitivity analysis of the model was performed, to determine how changes of the empirically estimated parameters affect the result of the calibration.

A preliminary analysis of 213 papers in the Journal of Artificial Societies and Social Simulation (JASSS) since 2011 by Troitzsch [50] revealed that 19.2%

of them compare quantitative simulation results to quantitative empirical data, while another 17.4% discuss the necessity of such comparison. Our model belongs clearly to the current minority of papers published on agent-based social (or demographic) simulation as it relies on diverse data sets for calibration. In addition, it uses theories, and other models. Although to test the assumed decision-making mechanisms at micro level further efforts are required, e.g., controlled cognitive experiments [14], a lot went already into developing the model and substantiating the claims made. The rest of the paper will be on methods to make these efforts more easily accessible and, thus, assessable.

3 Domain specific modeling languages

To establish trust in a complex multi-level simulation model it is necessary to understand how the model works. Therefore, a thorough and accessible description of the model is needed. Grimm et al. [20] proposed a standardized protocol, the ODD protocol, for the structured textual description of agent-based simulation models. The protocol is widely adopted, and is also recommended for uploading agent-based models in model repositories such as the OpenABM model repository. The motivation for ODD has been that the implementation of agent-based models are often difficult to understand, as models are rather lengthy and burdened with simulation details which dilutes the essential mechanisms how a model works. Domain specific modeling languages are aimed at bridging a gap between documentation and implementation of the model, with the ultimate goal to provide an executable documentation. Practical expressiveness, i.e., how easy is it to specify a model of a domain in the language and can also more complex mechanisms be expressed, and succinctness are central requirements for the design of domain-specific modeling languages. Whereas the former is difficult to measure requiring dedicated user studies[32], an indication for the later is the used line of code.

ML3 is a domain-specific modeling language specifically designed to allow a succinct and understandable implementation of continuous-time agent-based models with dynamic social networks, such as the migration decision model [53].

The main entities of models implemented in ML3 are *agents*, which are interconnected via dynamic *links*. In Figure 2 the definition of the agent type `Person`, representing the migrants, and of all types of links between persons are shown. Persons are characterized by a set of typed attributes, e.g., their sex (line 2), their income (line 3), or the stage of the decision process they are currently in (line 5). In addition to the explicitly specified attribute, every agent has an attribute `age`, describing the age of the agent. The `age` attribute is automatically 0, when the agent is created, and updated when time passes. The link types define how the social network of persons is shaped. Every defined link has two directions, which are read from left to right, and from right to left respectively. The link definition in line 15 defines that every person can have zero or more persons as their children. Conversely, those children have exactly two parents. ML3 ensures that those two directions always match. E.g., if person *B* is added as a child of person A, A is automatically added as a parent of B. Similarly, the following two lines define the `partner` and `friends` link. Agents can not only describe persons, but also higher level model entities. In this model, two further agent types `Household` and `Address`, represent households and places. The association of persons to households and the neighborhood relationship of places are also realized with links.

```
Person(
   sex: {"m", "f"},
   income: real := 0,
   mortalityModifier: real := 1,
   migrationStage: {"not viable", "intention" , "planning", "preparation", "
        migrated", "exit"} := "not viable",
   migrationAttempts: int := 0,
   failedMigrationAttempts: int := 0,
   status: {"child", "adult", "retired"} := "child",
   canAffordMonths: int := 0,
   canNotAffordMonths: int := 0,
   migrationAge: real := 0,
   migrationStartAge: real := 0
);

parents:Person[2] <-> [0-]Person:children;
partner:Person[0-1] <-> [0-1]Person:partner;
friends:Person[0-] <-> [0-]Person:friends;
```

Fig. 2: The ML3 declaration of the agent type `Person`, and the three possible kinds of links between persons.

Together, the agent type and link definitions describe how a state of the model looks like, i.e., what kinds of agents there are, and how they are interconnected. In addition to that, we have define how the model state evolves, i.e., how the agents act. In ML3, the behavior of agents is governed by stochastic *rules*, which are described as *guard-rate-effect* triples. In Figure 3 the rule that governs fertility is shown as an example. It describes that women in a certain age range may get a child. The first part of the rule is the *guard*, in line 2 beginning with a vertical bar. It specifies, to whom the rule applies, by giving a list of conditions the agent must fulfill. Agents are only subject to a rule, when they fulfill the guard condition. In this case, persons whose `sex` attribute has the value `"f"`, for female, and whose age is in a range defined by the constants `minFertilityAge` and `maxFertilityAge`. Here, the keyword `ego` refers to the agent for whom the guard is evaluated, similar to `this` in many object-oriented programming languages. The second part of the rule, the *rate*, is denoted in line 3. It specifies, when the rule is applied. The value of the rate expression defines the (possibly time-dependent, as the rate might depend of the agent's age or the global time) rate parameter of an exponential distribution. From this distribution a waiting time for every pair of agent and applicable rule is drawn. The agent-rule pair with minimal waiting time is then executed. Here the rate is defined through a function that encapsulates the rater complex calculation of a womens current stochastic rate of childbirth depending on her age and the number of children she already has. That way, the rule itself remain succinct, which makes its general function (describing childbirth) easier to read, while the details can be looked up if necessary. Finally, the third part of the rule, the *effect* (line 4 - 10), describes what happens, when the rule is executed. In this case, a new agent is created, it's attributes and links are initialized, and it is assigned a household and a place where it lives.

```
1  Person
2  | ego.sex = "f", ego.age >= minFertilityAge, ego.age < (maxFertilityAge + 1)
3  @ ego.birthRate()
4  -> ?child := new Person(
5         sex := ["m", "f"].random(),
6         parents := ego + if ego.isMarried() then [ego.partner] else [],
7         migrationStartAge := normal(meanMigrationStartAge, 1)
8       ),
9       ?child.moveToHousehold(ego.household),
10      ?child.moveToAddress(ego.address);
```

Fig. 3: The ML3 rule that describes the fertility component of the model. Female persons in a certain age range might get a child.

In comparison with the original model implementation using NetLogo, the implementation in ML3 has several advantages. Firstly, model and simulation algorithm are strictly separated. In ML3, the modeler only has to define stochastic rules. The actual scheduling, the selection when which rule is applied, is done by a separate simulator, that has been implemented by the developers of the language. In the NetLogo implementation, this scheduling had to be done manually, using a continuous time extension [47]. This separation of concerns makes the model description much more succinct (about a seventh of the length) and readable. As less implementation has to be written, there is less room for errors. Also, it makes the scheduling logic reusable, allowing to put more effort in implementing and testing simulation algorithms, enabling the implementation of more efficient advanced simulation algorithms [43].

Separation of concerns does not only apply to model and simulation logic, it also applies to the different components of the model. The stochastic rules that describe behavior in ML3 operate separate from each other in parallel. The different model components that are concerned with different processes operating in parallel, e.g., the fertility component (Figure 3) and the migration decision process (Figure 4), can be implemented as separate sets of rules. The separation of conceptually separate components is easily uphold in the implementation. Not only does this again contribute to an easier identification of mechanisms how the model works, it is also feasible to exchange component models by simply exchanging the corresponding rules. That way one could, for example, compare different decision process components in a multi-model approach, as proposed by Gray et al. [19]. Additionally, the stochastic rules allowed us to implement the transition through the stages of the transition process very naturally. The conceptual model of the decision process (Figure 1) defines stages an agent can occupy, and the transitions to other stages they can make from each of those. Our implementation (Figure 4) reflects this structure very directly, as it consists of one rule for every possible kind of transition.

The aspects of ML3 that make it domain-specific to the area of demography also play a major role in making the model description more succinct while enabling an easier implementation. For this model, two of them were of particular importance: the possible time dependent transition rates and parameter maps. Many of the transition rates in demographic models depend on the age of the agent. For example, mortality is strongly age-dependent, as are fertility and marriage. ML3 allows such time-dependency in transition rates and makes the aging of agents directly part of the language, at a cost to the efficiency of the execution due to the additional complexity in scheduling events. Languages not specific to this domain might not make this tradeoff. Parameter maps are a way to deal with time series data, which is also used excessively in this and other demographic models. The example shows the power of domain-specific modeling languages for assessing the structural validity of models and their role in modeling for explanation [14].

```
Person
| ego.inMigrationProcess(), ego.migrationIntention() >= 0, ego.migrationStage
     != "preparation"
@ ego.migrationAdvancementRate()
-> ego.advanceMigrationStage();

| ego.inMigrationProcess(), ego.migrationIntention() >= 0, ego.migrationStage =
      "preparation", ego.canAffordMigration()
@ ego.migrationAdvancementRate() * ego.borderEnforcementFactor()
-> ego.migrate();

Person.migrationAdvancementRate() := ?rho * e^(?a7 * ego.migrationIntention())
where ?rho := advancementRateBaseline,
      ?a7 := advancementRateIntentionWeight;

Person.migrationIntention() := ?a4 * ?MA + ?a5 * ?SN + ?a6 * ?PBC
where ?a4 := attitudeWeight,
      ?a5 := socialNormsWeight,
      ?a6 := perceivedBehavioralControlWeight,
      ?MA := ego.migrationAttitude(),
      ?SN := ego.socialNorms(),
      ?PBC := ego.perceivedBehavioralControl();
```

Fig. 4: Two of the ML3 rules concerned with the migration decision process. Each rule corresponds with one arrow, or kind of equivalent arrows, in Figure 1. The first rule corresponds the three arrows labeled with "waiting time expired". The second rule describes a successful migration attempt. Below that the definition of the rate with which agents progress through the stages of the decision process, and for calculation the migration intention based in the Theory of Planned Behavior are shown.

4 Explicitly specified experiments

To gain confidence in the validity of a multi-level demographic model it is necessary to probe and explore its behavior thoroughly. Therefore, diverse simulation experiments are required [30,34]. The results of simulation experiments, however, depend not only on the model, but also on the context in which they are executed. This insight has led to formalizations of this context, most prominently Zeigler's *experimental frame* [55]. Zeigler proposes to embed the model in an experimental frame, explicitly described as DEVS models, that generates the model inputs and analyzes the model outputs.

By enabling model users to access and repeat the simulation experiments conducted to validate a model, their confidence in the model's validity is increased. However, the two most frequently used ways to provide information about simulation experiments do not facilitate assessing the experiments done. First, executed experiments can be described informally, e.g. textually, or semi-formally as proposed in [20,21]. Repeating the experiments is typically hindered by unambiguous or missing information or unavailable software or data. A better approach is to provide software artifacts to execute the experiments. However, such executable software is typically hard or even impossible to inspect, leaving unclear what experiment is getting executed by it. Additionally, technical issues, such as dependencies on third-party software, makes running experiments difficult (or even impossible, for example if the dependencies are not available anymore some time after the experiments have been published). To address the challenges of accessing, repeating, and thus assessing simulation experiments, explicitly specified simulation experiments that allow the replication of experiments are needed.

SED-ML [52] has been developed in the SBML ecosystem in systems biology, originally to replicate published outputs of simulation experiments. Based on XML and cultivated by a standardization committee, SED-ML can be processed by many tools. These tools can interpret specifications with the standardized syntax and semantics, which enables tool-independent reproducible experiments. This makes SED-ML an effective exchange format for experiments. However, due to being a standard, new features can not easily be introduced in SED-ML. For example, parameter sweeps were not supported in the first release, but introduced in Level 1 Version 2 [5].

The Simulation Experiment Specification on a Scala Layer (SESSL) [18] aims to mitigate this lack of flexibility. SESSL is an internal domain-specific language, which enables on-the-fly addition of features through its host language Scala, for example to process simulation output data for further analysis [40]. The resulting experiment specifications are valid Scala code with a declarative feel. Thus, SESSL experiments are readable as well as executable. Further, by using Maven (`https://maven.apache.org`) for artifact persistence and management, SESSL experiments can be reproduced across machines. This way, model users can access and repeat validation experiments more easily.

Before a model can be validated, values for its input parameters must be found—the model is *calibrated*. As multi-level models typically contain parameters for speculative micro-level hypotheses, for which usually no validation data is available, this is a significant step towards a valid model. The migration model, for example, has eight weighting parameters that control the decision process of individuals. To find valid parameterizations of the model, methods to optimize a certain quality criterion can be employed, for example to minimize the difference between the model output and a given observation. Choosing a metric to calculate the difference is not a trivial task and depends heavily on the data to compare. Similarly, diverse optimization algorithms are available. Typically, these have to be parameterized as well. To make such a calibration experiment accessible and replicable, this information has to be included. Figures 5 and 6 shows how this can be realized with SESSL and its bindings for ML3 and Opt4j [36].

```
 1 class MigrationExperiment extends Experiment with ParallelExecution with
       ParameterMaps {
 2   model = "migration.ml3"
 3   simulator = NextReactionMethod()
 4   parallelThreads = -1
 5   replications = 1
 6   initializeWith(new JsonStateBuilder("initialstate.json"))
 7   startTime = 1982
 8   stopTime = 2050
 9
10   fromFile("maleMortality.csv")()
11   fromFile("femaleMortality.csv")()
12   fromFile("fertility.csv")()
13   fromFile("income.csv")()
14   fromFile("ageDifferenceModifier.csv")()
15   fromFile("baseMarriageRate.csv")()
16   fromFile("borderEnforcement.csv")()
17   fromFile("disc.csv")()
18
19   set("minFertilityAge" <~ 12, "maxFertilityAge" <~ 49)
20   set("ageOfAdulthood" <~ 16, "ageOfRetirement" <~ 65)
21   set("minMarriageAge" <~ 9, "maxMarriageAge" <~ 60)
22   set("meanMigrationStartAge" <~ 17)
23   set("spouseAgeModifier" <~ -0.01301431)
24   set("intercept" <~ -0.490129556)
25   set("homeCountryGini" <~ 0.4, "hostCountryGini" <~ 0.3)
26 }
```

Fig. 5: An experiment with the migration model defined in SESSL 0.14. Line 1 declares a Scala class that represents a ML3 simulation experiment that uses parallel execution and parameterization of models with parameter maps (e.g., age-indexed). Lines 2–8 specify the model file, the simulation algorithm, the number of parallel threads to use, the number of replications to execute, the method to build the initial state as well as the start and stop time of the simulation. In lines 10–17, files from which to read in parameters that are stored in CSV files with parameter maps are stated. Lines 19–25 specify some further, scalar model parameters.

```
 1 val referenceMean = 2
 2 
 3 minimize { (params, objective) =>
 4   execute {
 5     new MigrationExperiment with Observation {
 6       set("incomeEvaluationConstant" <~ params("a1"))
 7       set("incomeEvaluationCapitalWeight" <~ params("a2"))
 8       set("familyEvaluationWeight" <~ params("a3"))
 9       set("attitudeWeight" <~ params("a4"))
10       set("socialNormsWeight" <~ params("a5"))
11       set("perceivedBehavioralControlWeight" <~ params("a6"))
12       set("advancementRateIntentionWeight" <~ params("a7"))
13       set("advancementRateBaseline" <~ params("rho"))
14 
15       observeAt(Change(agentType = "Person", field = "migrationStage",
16         filter = "ego.migrationStage = 'migrated' && ego.planningTime != 0")) {
17         observe("decisionDuration" ~ expression("ego.planningTime"))
18       }
19 
20       var ages = List.empty[Double]
21 
22       withRunResult { result =>
23         if (result ? "decisionDuration")
24           ages ++= result.values("decisionDuration").asInstanceOf[Iterable[
                  Double]]
25       }
26 
27       withReplicationsResult { result =>
28         val mean = if (ages.nonEmpty) math.abs(referenceMean - (ages.sum / ages
                .size))
29                    else               Double.PositiveInfinity
30         objective <~ mean
31       }
32     }
33   }
34 } using new Opt4JSetup {
35   param("a1", 100.0, 5, 300)
36   param("a2", 0.0, 0.1, 2)
37   param("a3", 0.0, 5, 200)
38   param("a4", 0.0, 0.1, 2)
39   param("a5", 0.0, 0.1, 2)
40   param("a6", 0.0, 0.1, 2)
41   param("a7", 0.0, 1E-5, 1E-4)
42   param("rho", 0.0, 0.01, 1)
43   optimizer = ParticleSwarmOptimization(particles = 10, iterations = 20)
44 
45   withOptimizationResults { results =>
46     println("Overall results: " + results.head)
47   }
48 }
```

Fig. 6: A calibration experiment defined in SESSL 0.14, using the migration experiment class from Figure 5. The experiment uses a particle swarm optimization algorithm (line 43) from the Opt4j package [36] that tunes the parameters specified with ranges in lines 35–42. The parameter ranges can be chosen based on model assumptions, but here we set them based on the known optimum [28]. Lines 6–13 then read the parameters set by the optimizer and apply them to the model when running simulations. The observation of the model is configured in lines 15–18: the time between starting to plan a migration and actually migrating is recorded for every agent that actively migrates, i.e., is not brought with another migrating agent. The calculation of the target function to minimize is realized in lines 20–31. Specifically, line 20 declares a variable in which the planning times observed in single runs are stored (line 24). Lines 28–30 aggregate the durations to the mean and compute the difference to a reference mean (specified in line 1).

Once calibrated, behavioral characteristics of the model can be checked by simulating the model and making sure that the observations from the simulation match some defined expectations, for example derived from data. A formal framework for this approach is *statistical model checking* (SMC) or simulation-based verification. SMC answers the question whether a random simulation run of a model satisfies a given property with at least a given probability [33]. Applying SMC to a model implies executing simulation runs, checking the property on each of the runs, and using hypothesis testing to infer statistically valid statements about the model's behavior. Thus, the properties to investigate must be defined on model outputs that are observable in a simulation run. Typically, temporal logics are used to express statements about the development of the model outputs in time. SMC experiments can be specified reproducibly by including the property to check as well as the statistical parameters in the experiment set-up.

Apart from calibration and statistical model checking, many more types of experiments are required to validate a model. To make such experiments accessible, replicable and, thus, assessible, experiment specification languages need to support a wide variety of experiments, the set of which might be constantly growing. Particularly the later poses a challenge for the design and development of these languages, as they must satisfy constantly changing requirements while supporting a succinct and understandable description of experiments.

5 Provenance models for dependencies made explicit

Whereas the above domain-specific languages allow the user to succinctly specify and execute a model as well as a simulation experiment, what [42] requested in his *Preferred Model Reporting Requirements* (PMRR) to include in addition, i.e., information on the sources of data for the model's equations and algorithmic rules, has not been considered yet. However, given that not only data as input and data for calibration, but also theories, such as the Theory of Planned Behavior [1], and existing models such as the Heligman-Pollard mortality model [24], and, ideally, cognitive controlled experiments to probe the micro level decision mechanisms, contribute to a multi-level model in demography, only focusing on data will not suffice. Therefore, a more systematic inspection of a simulation model's provenance is asked for [44].

"Provenance is 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"[22]. The Open Provenance Model (OPM) [38] allows to describe this provenance information as a directed graph, where nodes represent *artifacts*, *processes*, or *agents*, while edges indicate dependencies. Here, *artifacts* are digital representations of entities within a computer system, in our case component models, data sources or experiment specifications. *Processes* represent activities performed with artifacts to generate new artifacts. And *agents* are the entities enabling and controlling the processes. Between these elements five dependencys are distinguished: 1. an artifact was *used* in a process,

2. an artifact *was generated by* a process, 3. a process *was controlled by* an agent, 4. a process *was triggered by* another process, and 5. an artifact *was derived from* another artifact.

To demonstrate our approach presented in [44], we tried to reconstruct the provenance information about the migration decision process model from the publication about the model [28], the ODD description [26], and the information provided together with the model in the OpenABM model repository [27]. We focus on the information about artifacts and processes, and leave out information about agents, as we have little information about who exactly was involved each process. The result, which is by no means complete, due to the limited amount of information we have, is shown in Figure 7. The migration decision process model itself (*mig. model*) is shown as an artifact on left side of the figure. It was produced through composing its models components (process *comp. model*). We will now look at one of the components, the mortality model, in detail. The mortaliy model (artifact *fitted HP*) it is directly derived from the established Heligman-Pollard model of mortality (artifact *HP*), as the fitted model is just a variant of the unfitted model, with concrete values set for its parameters. It was produced through the process of fitting the model (process *fit HP*) to the UN World Population Prospects 2015 data on mortality in Senegal (artifact *WPP*).

In the provenance model the dependencies between the processes and artifact, and their interdependencies, become explicit. This explicitness can be used to improve our trust into the model. We have now made it explicit, that the mortality component of the model is derived from the Heligman-Pollard model. That model is widely applied and its validity for different applications has been assessed [10]. At the same time, the provenance model tell us, which data was used to fit the Heligman-Pollard model to the Senegal case. But the provenance model does not only consider the artifacts, but also the processes through which they were derived. It makes the process of fitting explicit, pointing the modeler to the need to document it, and making a later reader aware of it. All together, this information gives us trust in the mortality component of the migration model. In general, the provenance information enables us to reconstruct assumptions made in the components, the theories they were derived from, methods used for developing them, and data sources used for fitting the components and trace them to their origin. Further we can use provenance data to reason about the origins used and processes executed for the development the component models. For example if we identify an methodological error in the collection of a dataset we can use the provenance model to infer affected model components which need to be revised. For this, inference mechanisms such as OPQL [35] can be employed.

Similar to the provenance of the simulation model, we can document the provenance of simulation experiments. We have done so for the calibration experiment executed by Klabunde et al., and show the result in Figure 8. In the figure, the migration process model artifact (*mig. model*) is shown once again. The description of the calibration experiment is shown as another artifact (*calibr. exp*). The process of executing the calibration experiment (*run calibr.*) uses those two artifacts, as well as the target value of the calibration (artifact *plan. time*)

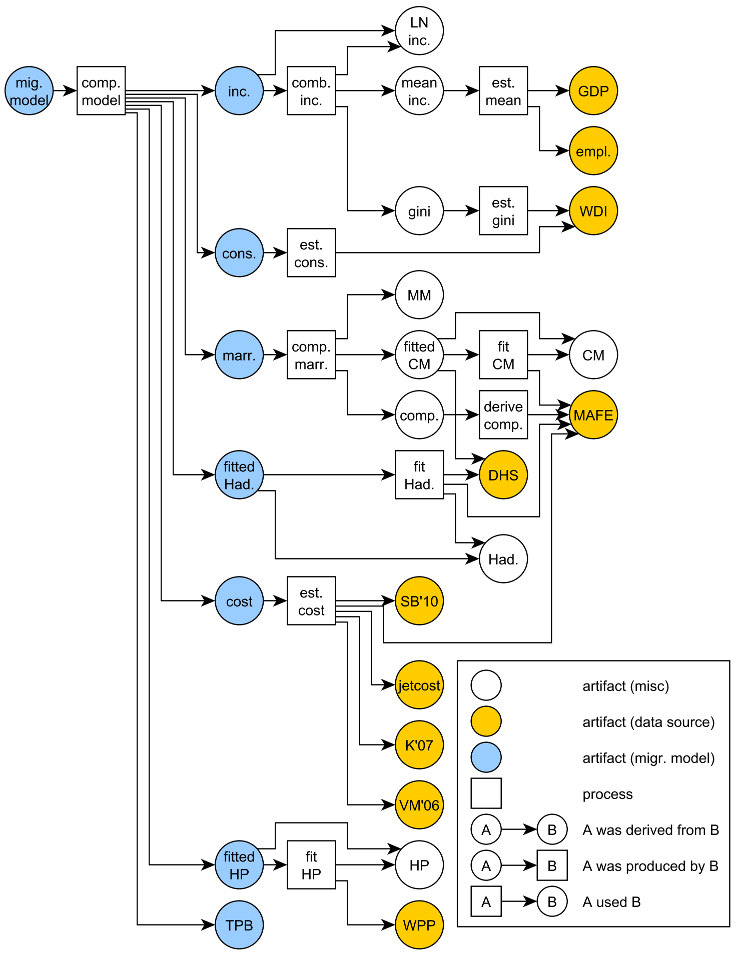

Fig. 7: Open Provenance Model for the migration decision process model, as we have derived it from the publications about the model. The artifacts and processes in this figure are shortly described in Tables 1, 2, 3, and 4.

and the initial artificial population (*init. pop*). The latter two are derived from different sources of data through various processes. The result of the calibration process is a calibrated model (artifact *calibr. model*). It is directly derived from the original migration model, as it only differs from it in the parameter values set.

Provenance information about the simulation experiment can be used, similar to provenance of the model, to trace the origin of data and methods used in the experiment. The origin of the data is especially of interest for validation and

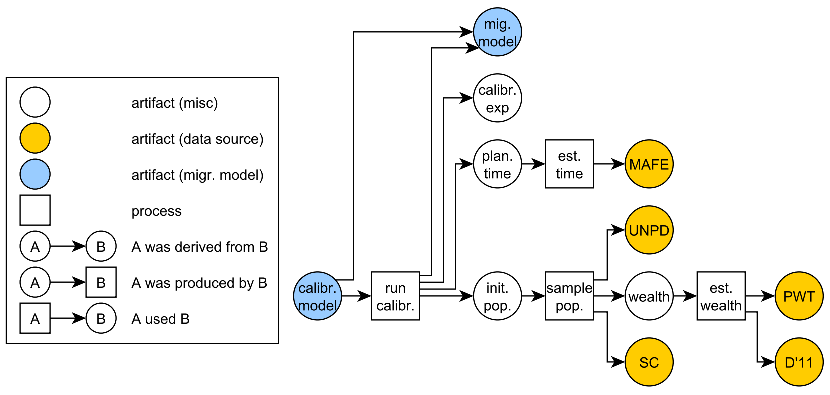

Fig. 8: Open Provenance Model for the calibration experiment. Note that the artifacts *mig. model* and *MAFE* are identical to the artifacts of the same name in Figure 7. The artifacts and processes in this figure are shortly described in Tables 1, 2, 3, and 4.

calibration experiments, because if the same data used for the calibration of the simulation is also used for validation of the simulation model the validation result would be invalid. Further, we can use the provenance information to find all artifacts used for the execution of an experiment. This information can be used to bundle these artifacts into a container which contains all the data sources, the simulation model and the simulation experiment specification. The container can be then shared and therefore allow to replicate the experiment result.

The reconstruction of the provenance data by tracing publications using the model documentation is cumbersome and therefore (semi-)automatic methods for retrieving provenance data are needed. For this, three different techniques were proposed by Moreau et al. [38], where each method allows to retrieve different parts of the provenance information.

Firstly, we can use ontologies to annotate parts of the simulation model and experiments which allows to specify all information related to the process of creating the simulation model or executing the simulation experiment. An ontology allows describing entities, relations between entities and attributes of entities in a domain-specific vocabulary. For example, we can annotate the migration model (*mig. model*) with information about the model components used e.g., the fitted Heligman-Pollard model (*fit. HP*). By annotating the fitted Heligman-Pollard model we can describe information about the fitting process like the used data source. To derive the provenance information from these annotations, the annotations need to be parsed and analyzed. This approach demands a close annotation of all artifacts to derive the full provenance model. However, it is up to the user to annotate the simulation model and experiments as part of the documentation and therefore can only be seen as a semi-automatic approach. Please note, the

Open Provenance Model itself can be described using an ontology, which allows to provide additional information about the artifacts and dependencies using further ontologies [38].

When scrips, scientific workflow, or domain specific languages are employed to execute automatic tasks, those can also be used as source of provenance information, by integrating the derivation of provenance information into them. All these approaches allow to describe data-driven processes, but differ in the way the process is described and which features are provided by the execution environment. In scripting environments, like Python or R, the script is executed by the environment and it is up to the user to implement management procedures for the data by himself. Scientific workflows are often described by a graph containing nodes representing data-processing activities whereas the edges represent the flow of the data. These environments are especially designed for data-driven processes and therefore provides a rich feature sets of common used methods for data processing. Finally, domain specific languages can be used to describe a simulation experiments, as we have demonstrated in Section 4. These languages are designed to describe a simulation experiment which can be executed in an appropriate environment which uses data management methods without explicit description required by the user. However, as these approaches focus the execution of processes only provenance data related to the processes can be collected. As a result, the description of the process itself becomes an artifact and the execution of the process becomes a process in the provenance model. In case of scripts and domain specific languages two different methods seems suitable to retrieve the related artifacts. Firstly, the execution of the script or created simulation experiment can be observed to determine all read and created files [39], which become individual artifacts in the provenance model. And secondly, the script itself can call methods to store provenance information [9]. Scientific workflow environments often provide features to retrieve the provenance information directly after the execution of the workflow [45].

Finally, a version control system can be used to derive provenance information by tracking changes to document. For example if the migration model artifact is stored in an document we can track how the model changes over time. However, the version control system can only capture the changes to artifacts, but not the processes that produce the changes. Therefore we only get the *was derived from* dependency from the version control system. For example if the simulation model was changed during the calibration process this change can be recognized and allows us to retrieve the *was derived from* dependency, but we can not get further information like the target values about the calibration process. This can supported using tools like Git2PROV [16], which retrieves a provenance model from the commit history of a git repository.

6 Conclusion

The validation of multi-level models in demography provides many challenges, as we speculate at micro level and calibrate at macro level. Ideally, the validity of a

Id.	Description	Ref.
calibr. model	calibrated migration decision process model	[28]
cons.	consumption time series for Senegal and France	[27]
cost	migration cost model	[27]
fitted Had.	fitted Hadwiger model	[27]
fitted HP	fitted Heligman-Pollard model	[27]
inc.	lognormal income distributions for Senegal and France at multiple times	[27]
marr.	marriage model component	[27]
mig. model	migration decision process model	[28]
TPB	Theory of Planned Behavior as model of a decision process	[1]

Table 1: Open Provenance Model artifacts corresponding to the migration decision process model and its components.

Id.	Description	Ref.
D'11	Davies et al. about the level and distribution of global household wealth	[15]
DHS	Senegal Demographic and Health Survey 1986 - 2014	
empl.	IMF employment data	
GDP	IMF GDP data	
jetcost	flight cost data retrieved from jetcost.de (specifics unknown)	
K'07	estimations of the cost of migration by boat from Senegal to the Canary Islands by Kohnert	[31]
MAFE	Migration from Afrika to Europe	[4]
PWT	Penn World Table 6.1	
SB'10	estimation of the cost for smugglers for illegal migration by plane and ship by Schmid and Borchers	[46]
SC	Senegal Census 1988	
UNPD	UN Population Division data about the age structure of Senegalese in France in 1982	
VM'06	estimations of the cost of the migration by boat from Senegal to various points in Europe by van Moppes	[51]
WDI	World Development Indicators	
WPP	UN World Population Prospects 2015	

Table 2: Open Provenance Model artifacts represent sources of data.

Id.	**Description**	**Ref.**
calibr. exp	calibration experiment	[28]
CM	Coale-McNeil model of transition rates to marriage	[12]
comp.	partner compatibility measure to use by the marriage market	[26]
gini	Gini index estimation for Senegal and France	[27]
Had.	Hadwiger model of fertility rates	[23]
HP	Heligman-Pollard model of age-dependent force of mortality	[24]
init pop.	initial population for the simulation	[27]
LN inc.	lognormal distribution as model for income distributions	[3,11]
mean inc.	mean income time series for Senegal and France	[27]
MM	marriage market for matchmaking	[56]
plan. time	mean time from starting to plan migration to the actual migration	[27]
wealth	estimated distribution of household wealth in Senegal	[27]

Table 3: Open Provenance Model artifacts that are neither sources of data, nor corresponding to the migration decision process model and its components.

Id.	**Description**	**Ref.**
comb. inc.	combination of the estimated income measures to derive estimated income distributions for Senegal and France	[27]
comp. marr.	composition of the different components of the marriage model	[27,28]
comp. model	composition of the different components of the migration decision model	[28]
derive comp.	derivation of a compatibility measure from the MAFE data	[26]
est cons.	estimation of a consumption time series	[27]
est. cost	estimation of the mean migration cost as a weighted average of the migration cost when using different modes of transit	[27]
est. gini	estimation of Gini indices in Senegal and France	[27]
est. mean.	estimation of a mean income time series in Senegal and France	[27]
est. time	estimation of the mean time from starting to plan migration to actually migrating	[27]
est. wealth	estimation of the wealth distribution for Senegalese households	[27]
fit CM	fitting of the Coale-McNeil model to the Senegal data	[27,28]
fit Had.	fitting of the Hadwiger model to the Senegal data	[27]
fit HP	fitting of the Heligman-Pollard model to the Senegal data	[27]
run calibr.	execution of the calibration experiment	[28]
sample pop.	sampling of the initial population for the simulation	[27]

Table 4: Open Provenance Model processes.

micro model, e.g., the diverse decisions that are involved in determining whether or not to migrate, is supported by controlled cognitive experiments. However, even in the absence of such experiments, evidence for the validity of a multi-level model can be found in a multitude of artifacts. Based on a migration model, we illustrate how making these artifacts and their relations more easily accessible contributes to increasing the trust in the model. The key ingredients in this endeavor are declarative representations that can be executed or reasoned about. Domain specific modeling language provide an executable, yet succinct model representation that reflects the structure of the conceptual model, facilitating a better understanding of a model's mechanisms. Domain-specific languages for specifying experiments support not only the documentation, but also the replication of simulation experiments, and thus give insight into a model's behavioral repertoire. Crucial at this point are the flexibility and extensibility of the languages to account for the increasing number of simulation methods that, for example, a demographic multi-level model should be subjected to. Methods such as statistical model checking allow simulation experiments to explicitly specify expectations for the behavior of the model, which is a step towards reducing the need for human interaction. Finally, exploiting the provenance model allows to relate the diverse artifacts that contributed to a simulation model and the process of simulation experimentation. While this is already encouraged by widely used protocols such as ODD, the formalization in a provenance model allows to partly automate the process and reason about the relations of artifacts using inference mechanisms. This way, experiment specification, theories that underlie the model, as well as data used as input or for calibration can be linked even beyond individual simulation studies. All together reveal not the whole, but a quite fascinating tale about the science and art of model developing.

Acknowledgements. This research was supported by the German Research Foundation (DFG) via research grant UH-66/15 and research grant UH-66/18.

References

1. I. Ajzen. The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, 50(2):179–211, Dec. 1991.
2. O. Balci. Verification, validation and accreditation of simulation models. In *Proceedings of the 29th conference on Winter simulation, WSC 1997, Atlanta, GA, USA, December 7-10, 1997*, pages 135–141. ACM, 1997.
3. R. Bandourian, J. McDonald, and R. S. Turley. A Comparison of Parametric Models of Income Distribution Across Countries and Over Time. SSRN Scholarly Paper ID 324900, Social Science Research Network, Rochester, NY, June 2002.
4. C. Beauchemin. Migration between Africa and Europe (MAFE): Looking beyond Immigration to Understand International Migration. *Population*, 70(1):13–38, 2015.
5. F. T. Bergmann, J. Cooper, N. Le Novere, D. Nickerson, and D. Waltemath. Simulation experiment description markup language (SED-ML) Level 1 Version 2. *Journal of Integrative Bioinformatics (JIB)*, 12(2):119–212, 2015.

6. J. Bijak, J. Hilton, E. Silverman, and V. D. Cao. Reforging the Wedding Ring: Exploring a Semi-Artificial Model of Population for the United Kingdom with Gaussian process emulators. *Demographic Research*, 29(27):729–766, Oct. 2013.
7. F. Billari, B. Aparicio Diaz, T. Fent, and A. Fürnkranz-Prskawetz. The "Wedding-Ring": An agent-based marriage model based on social interaction. *Demographic Research*, 17(3):59–82, Aug. 2007.
8. F. C. Billari. Integrating macro- and micro-level approaches in the explanation of population change. *Population Studies*, 69(Suppl.):S11–S20, 2015.
9. C. Bochner, R. Gude, and A. Schreiber. A python library for provenance recording and querying. *Provenance and Annotation of Data and Processes*, pages 229–240, 2008.
10. H. Booth and L. Tickle. Mortality Modelling and Forecasting: A Review of Methods. *Annals of Actuarial Science*, 3(1-2):3–43, Sept. 2008.
11. D. Chotikapanich, W. E. Griffiths, and D. S. P. Rao. Estimating and Combining National Income Distributions Using Limited Data. *Journal of Business & Economic Statistics*, 25(1):97–109, Jan. 2007.
12. A. J. Coale and D. R. Mcneil. The distribution by age of the frequency of first marriage in a female cohort. *Journal of the American Statistical Association*, 67(340):743–749, Dec. 1972.
13. J. S. Coleman. Social theory, social research, and a theory of action. *American journal of Sociology*, 91(6):1309–1335, 1986.
14. R. Conte, N. Gilbert, G. Bonelli, C. Cioffi-Revilla, G. Deffuant, J. Kertesz, V. Loreto, S. Moat, J.-P. Nadal, A. Sanchez, et al. Manifesto of computational social science. *European Physical Journal-Special Topics*, 214:p–325, 2012.
15. J. B. Davies, S. Sandström, A. B. Shorrocks, and E. N. Wolff. The Level and Distribution of Global Household Wealth. *The Economic Journal*, 121(551):223–254, Mar. 2011.
16. T. De Nies, S. Magliacane, R. Verborgh, S. Coppens, P. Groth, E. Mannens, and R. Van de Walle. Git2prov: exposing version control system content as w3c prov. In *Proceedings of the 2013th International Conference on Posters & Demonstrations Track-Volume 1035*, pages 125–128. CEUR-WS. org, 2013.
17. B. A. Diaz, T. Fent, A. Prskawetz, and L. Bernardi. Transition to parenthood: The role of social interaction and endogenous networks. *Demography*, 48(2):559–579, 2011.
18. R. Ewald and A. M. Uhrmacher. SESSL: A Domain-specific Language for Simulation Experiments. *ACM Trans. Model. Comput. Simul.*, 24(2):11:1–11:25, 2014.
19. J. Gray, J. Hilton, and J. Bijak. Choosing the Choice: Reflections on modelling decisions and behaviour in demographic agent-based models. *Population Studies*, 71, May 2017.
20. V. Grimm, U. Berger, F. Bastiansen, S. Eliassen, V. Ginot, J. Giske, J. Goss-Custard, T. Grand, S. K. Heinz, G. Huse, A. Huth, J. U. Jepsen, C. Jørgensen, W. M. Mooij, B. Müller, G. Pe'er, C. Piou, S. F. Railsback, A. M. Robbins, M. M. Robbins, E. Rossmanith, N. Rüger, E. Strand, S. Souissi, R. A. Stillman, R. Vabø, U. Visser, and D. L. DeAngelis. A standard protocol for describing individual-based and agent-based models. *Ecological Modelling*, 198(1):115 – 126, 2006.
21. V. Grimm, U. Berger, D. L. DeAngelis, J. G. Polhill, J. Giske, and S. F. Railsback. The ODD protocol: A review and first update. *Ecological Modelling*, 221(23):2760 – 2768, 2010.
22. P. Groth and L. Moreau. PROV-overview. An overview of the PROV family of documents. 2013.

23. H. Hadwiger. Eine analytische Reproduktionsfunktion für biologische Gesamtheiten. *Scandinavian Actuarial Journal*, 1940(3-4):101–113, July 1940.
24. L. Heligman and J. H. Pollard. The age pattern of mortality. *Journal of the Institute of Actuaries*, 107(1):49–80, Jan. 1980.
25. A. Klabunde and F. Willekens. Decision-Making in Agent-Based Models of Migration: State of the Art and Challenges. *European Journal of Population*, 32(1):73–97, Feb. 2016.
26. A. Klabunde, F. Willekens, S. Zinn, and M. Leuchter. An agent-based decision model of migration,embedded in the life course – Model description in ODD+D format. Technical report, Max Planck Institute for Demographic Research, Rostock, Germany, June 2015.
27. A. Klabunde, S. Zinn, F. Willekens, and Leuchter. Multistate modeling extended by behavioral rules (Version 6). https://www.openabm.org/model/5146/version/6/view, Aug. 2016.
28. A. Klabunde, S. Zinn, F. Willekens, and M. Leuchter. Multistate modeling extended by behavioral rules - an example of migration. *Population Studies*, 2017.
29. J. P. Kleijnen. An overview of the design and analysis of simulation experiments for sensitivity analysis. *European Journal of Operational Research*, 164(2):287–300, 2005.
30. F. Klügl. A Validation Methodology for Agent-based Simulations. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 39–43, New York, NY, USA, 2008. ACM.
31. D. Kohnert. African Migration to Europe: Obscured Responsibilities and Common Misconceptions. *GIGA Working Paper*, 49, 2007.
32. C. Kossow, T. Helms, J. M. Kreutzer, A. Martens, and A. M. Uhrmacher. Evaluating Different Modeling Languages Based on a User Study. In *Proceedings of the 49th Annual Simulation Symposium*, ANSS '16, pages 18:1–18:8, San Diego, CA, USA, 2016. Society for Computer Simulation International.
33. A. Legay, B. Delahaye, and S. Bensalem. Statistical model checking: An overview. In *Proceedings of the First International Conference on Runtime Verification*, RV'10, pages 122–135, Berlin, Heidelberg, 2010. Springer-Verlag.
34. S. Leye, J. Himmelspach, and A. M. Uhrmacher. A Discussion on Experimental Model Validation. In *2009 11th International Conference on Computer Modelling and Simulation*, pages 161–167, Mar. 2009.
35. C. Lim, S. Lu, A. Chebotko, and F. Fotouhi. OPQL: A First OPM-Level Query Language for Scientific Workflow Provenance. pages 136–143. IEEE, July 2011.
36. M. Lukasiewycz, M. Glaß, F. Reimann, and J. Teich. Opt4J - A Modular Framework for Meta-heuristic Optimization. In *Proceedings of the Genetic and Evolutionary Computing Conference (GECCO 2011)*, pages 1723–1730, Dublin, Ireland, 2011.
37. C. Maus, S. Rybacki, and A. M. Uhrmacher. Rule-based multi-level modeling of cell biological systems. *BMC Systems Biology*, 5:166, Oct. 2011.
38. L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, et al. The open provenance model core specification (v1. 1). *Future generation computer systems*, 27(6):743–756, 2011.
39. L. Murta, V. Braganholo, F. Chirigati, D. Koop, and J. Freire. noworkflow: capturing and analyzing provenance of scripts. In *International Provenance and Annotation Workshop*, pages 71–83. Springer, 2014.
40. D. Peng, T. Warnke, F. Haack, and A. M. Uhrmacher. Reusing simulation experiment specifications to support developing models by successive extension. *Simulation Modelling Practice and Theory*, 68:33 – 53, 2016.

41. 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. *SIMULATION*, 93(8):659–677, 2017.
42. H. Rahmandad and J. D. Sterman. Reporting guidelines for simulation-based research in social sciences. *System Dynamics Review*, 28(4):396–411, 2012.
43. O. Reinhardt and A. M. Uhrmacher. An Efficient Simulation Algorithm for Continuous-time Agent-based Linked Lives Models. In *Proceedings of the 50th Annual Simulation Symposium*, ANSS '17, pages 9:1–9:12, San Diego, CA, USA, 2017. Society for Computer Simulation International.
44. A. Ruscheinski and A. M. Uhrmacher. Provenance in modeling and simulation studies-bridging gaps. 2017.
45. C. Scheidegger, D. Koop, E. Santos, H. Vo, S. Callahan, J. Freire, and C. Silva. Tackling the provenance challenge one layer at a time. *Concurrency and Computation: Practice and Experience*, 20(5):473–483, 2008.
46. S. Schmid and K. Borchers. *Vor den Toren Europas? Das Potenzial der Migration aus Afrika.* Number 7 in Forschungsbericht / Bundesamt für Migration und Flüchtlinge. Nürnberg, 2010.
47. C. J. Sheppard and S. Railsback. Time Extension for NetLogo (Version 1.2) [Software]. Available from https://github.com/colinsheppard/time, 2015.
48. W. M. Trochim and J. P. Donnelly. *The Research Method Knowledge Base.* Cengage Learning, 2008.
49. K. G. Troitzsch. Validating simulation models. In G. Horton, editor, *18th European Simulation Multiconference. Networked Simulations and Simulation Networks*, pages 265–270, 2004.
50. K. G. Troitzsch. Using Empirical Data for Designing, Calibrating and Validating Simulation Models. In *Advances in Social Simulation 2015*, Advances in Intelligent Systems and Computing, pages 413–427. Springer, Cham, 2017.
51. D. van Moppes. The African Migration Movement: Routes to Europe. Technical report, Nijmegen, Jan. 2006.
52. D. Waltemath, R. Adams, F. T. Bergmann, M. Hucka, F. Kolpakov, A. K. Miller, I. I. Moraru, D. Nickerson, S. Sahle, J. L. Snoep, and N. Le Novère. Reproducible computational biology experiments with SED-ML - The Simulation Experiment Description Markup Language. *BMC Systems Biology*, 5(1):198, Dec 2011.
53. 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. *Population Studies*, 2017.
54. R. G. Wiegert. Holism and reductionism in ecology: hypotheses, scale and systems models. *Oikos*, pages 267–269, 1988.
55. B. P. Zeigler, H. Praehofer, and T. G. Kim. *Theory of Modeling and Simulation.* Academic Press, San Diego, CA, USA, 2nd edition, 2000.
56. S. Zinn. A Mate-Matching Algorithm for Continuous-Time Microsimulation Models. *International Journal of Microsimulation*, 5(1):31–51, 2012.