Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

The Role of Languages for Modeling and Simulating Continuous-Time Multi-Level Models in Demography

Alexander Steiniger
Adelinde M. Uhrmacher

University of Rostock
Albert-Einstein-Str. 22
D-18059 Rostock, GERMANY

Sabine Zinn
Jutta Gampe
Frans Willekens

Max Planck Institute for Demographic Research
Konrad-Zuse-Str. 1
D-18057 Rostock, GERMANY

ABSTRACT

Demographic microsimulation often focuses on effects of stable macro constraints on isolated individual life course decisions rather than on effects of inter-individual interaction or macro-micro links. To change this, modeling and simulation have to face various challenges. A modeling language is required allowing a compact, succinct, and declarative description of demographic multi-level models. To clarify how such a modeling language could look like and to reveal essential features, an existing demographic multi-level model, i.e., the linked life model, will be realized in three different modeling approaches, i.e., ML-DEVS, ML-Rules, and attributed pi. The pros and cons of these approaches will be discussed and further requirements for the envisioned language identified. Not only for modeling but also for experimenting languages can play an important role in facilitating the specification, generation, and reproduction of experiments, which will be illuminated by defining experiments in the experiment specification language SESSL.

1 INTRODUCTION

In demography, in addition to macro-projection (Bohk et al. 2009), microsimulation plays an important role (Willekens 2005, Willekens 2009, van der Gaag 2009). Its essence is the individual's life-course, which is defined by the sequence of states that the individual visits over time and the waiting times between state transitions. The time-scales can either be discrete (usually in units of years) or continuous. In discrete-time models, time advances in discrete, equidistant steps. In contrast, a continuous-time microsimulation has a continuous time scale along which events can occur. In general, continuous-time models are the optimal theoretical choice for a precise description of population dynamics, as they mirror life-course developments most closely (Willekens 2005). The underlying stochastic model of continuous-time microsimulation is a continuous-time Markov model parameterized with transition rates. Accordingly, a continuous-time microsimulation model is also a competing risks model (cf. Zinn (2011)).

Currently, demographic microsimulation focuses often on effects of stable macro constraints on isolated individual life course decisions rather than on effects of inter-individual interaction (Jager and Janssen 2003) or the macro-micro link (Ewert et al. 2007, Gilbert and Troitzsch 2008). To change this, we have to face various challenges. Two issues seem to refer to modeling: (a) how to model inter-individual interactions and dynamic binding/grouping (e.g., household formation and dissolution) in continuous space and time and (b) how to integrate dynamics at different levels. Another challenge refers to validation, as with more expressive models, demography has started to move from data- to hypotheses-driven development of individual-based, mechanistic models. This has an impact on validation processes and methods used. The question how far already existing approaches address requirements of multi-level modeling and simulation in demography, we will explore based on the *linked lives* model of Noble et al. (2012).

2 LINKED LIVES

The *linked lives* model presented by Noble et al. (2012) is a discrete time-stepped, probabilistic agent-based (i.e., individual-based) model, which aims at predicting the supply of and demand for social care in the modern UK, based on basic demographic processes. These include mortality, fertility, health changes, migration, and the formation and dissolution of partnerships and households. A central idea of the model is that the lives of individuals can be *linked* (e.g., individuals can form partnerships) and that these links have an influence on individual life courses. The three central entities of the model are: *agents*, *houses*, and *towns*. These entities are nested such that agents live in houses that are located in different towns. At each time step in the simulation (yearly time scale), the agents get older and can, e.g., move around, marry (form a partnership), give birth (if the agent is female), get divorced, or die according to age-specific probabilities. Some of these actions trigger others. For instance, if a child loses both its parents, the child will be adopted. Combined, all actions define the transition of the state of the overall model. In contrast to the *Wedding Ring* model (Billari et al. 2007) and its extension the *Wedding Doughnut* (Silverman et al. 2012), both are different but well-known models in agent-based computational demography, partnerships are not permanent and the marriage market is not spatially constrained but global in the *linked lives* model. In addition, social pressure on unmarried individuals exerted by relevant others that are already married is not considered. The model was implemented in Python and the source code is freely available.

3 THE ROLE OF LANGUAGES FOR MODELING

The exploitation of agent-based approaches in computational demography leads to complex causal models of individuals' behavior (microscopic view) and with this to the desire for a compact description of those models. Often, the models are either implemented from scratch using a high-level programming language or realized based on specific agent-based tools. For instance, the *linked lives* model and its simulator are implemented in Python, whereas Silverman et al. (2012) use the multi-agent modeling and simulation framework *Repast Symphony* to implement the *Wedding Doghnut*. A plethora of such agent-based tools exist and are used (Nikolai and Madey 2009), but only a few support continuous-time models. Typically, predefined templates and classes are offered and high-level programming languages can be used as a host language to define individual agents, often supported by a GUI. All these approaches hinder enclosing and discussing realistic models in publications, as the models' descriptions are not sufficiently compact and seldom make use of the idioms at the level of abstraction of the problem domain. The latter may also make it more difficult to understand the models for a domain expert, who is not necessarily familiar with programming. Thus, a domain-specific modeling language is desirable. The question is how such a language for continuous-time multi-level models in demography could look like.

The role that a comprehensive, concise modeling language with an adequate expressiveness plays for simulation, has led to a variety of modeling languages for Markovian populations in computational systems biology over the last years, e.g., Faeder et al. (2009), John et al. (2010), or Maus et al. (2011). Some languages equip model entities (e.g., processes) with attributes and allow restricting reactions according to constraints on attributes of the reactants, e.g., John et al. (2010). Likewise, transitions of individuals from one state to another can be constraint by attributes of the involved individuals or households. Dynamic grouping or nesting (Krivine et al. 2008, Cardelli and Gardner 2010, Maus et al. 2011) is of interest for cell biological models, but also for demographic ones. For instance, individuals reside in different houses. The idea to assign species (model entities) to compartments, making activities dependent on the location, and having species diffuse or being transported between locations can also easily be adapted for demographic migration between discrete spatial regions. Similarly, dynamic bindings could directly be translated to *linked lives*. However, also differences to systems biology exist, e.g., referring to organizational structures, as the "being-part-of"-relation is not exclusive, but individuals can belong to different organizations (groups) at the same time. Thus, although some approaches may already address central requirements, e.g., variable structures or linking of individuals' and groups' dynamics in continuous time and discrete space, none

has been truly tailored for the task at hand. To analyze specific requirements of demographic multi-level models, we realize the *linked lives* model in three different modeling languages/approaches, i.e., ML-DEVS (Steiniger et al. 2012), ML-Rules (Maus et al. 2011), and attributed π (John et al. 2010).

3.1 ML-DEVS

The *Multi-Level Discrete Event System Specification* (ML-DEVS) is a modular, hierarchical modeling formalism for variable structure and multi-level modeling. As in other DEVS variants, a system of interest is described as a *reactive system* asynchronously interacting with its environment, which can be a reactive system itself. ML-DEVS distinguishes between Micro-DEVS and Macro-DEVS models (short micro and macro models), the latter of which can again contain micro and macro models as components. In addition to a flexible horizontal coupling of model components via *multi-couplings*, ML-DEVS allows an explicit *up-* and *downward causation* between different levels. The inter-level causation is, however, restricted to neighboring levels (micro and macro level). DEVS variants, such as ML-DEVS, provide a suitable framework for modeling reactive, autonomous agents (Kim and Kim 2001, Steiniger et al. 2012). Also different variants of DEVS (including a variant of ML-DEVS) have already been used for microsimulation in computational demography (Zinn et al. 2010, Zinn 2011). In contrast to Zinn (2011), the agents in our model directly interact via the multi-couplings.

To realize the *linked lives* model in ML-DEVS, the first approach that may come into mind is to reflect the organizational structure of the original model by the composition structure of the ML-DEVS model. Such a model would comprise the component: *agents*, *houses*, and *towns*, with agents being sub-component of houses that are sub-components of towns. However, this from a hierarchical modeling point of view intuitive approach is (although possible) rather impractical for the following reasons: (a) Up- and downward causation can only be defined between a macro model and the models at the macro model's micro level. If we have to cross several levels, we need to define the up- and downward causation for each pair of macro model and micro level, which is cumbersome and leads to redundancies in the model definition. (b) Even though ML-DEVS supports a variable model structure, the "migration" of model components (e.g., an agent moves from one house to another) is not directly supported, but has to be realized indirectly by introducing additional ports via which model components can be sent and by an additional logics that reacts to model components being sent. Instead, we propose a flat model hierarchy as depicted in Figure 1 (left), similarly to Zinn (2011), with only two levels (one macro and micro level). A central macro model (called Population) contains micro models representing the agents, which have unique names. The agents'

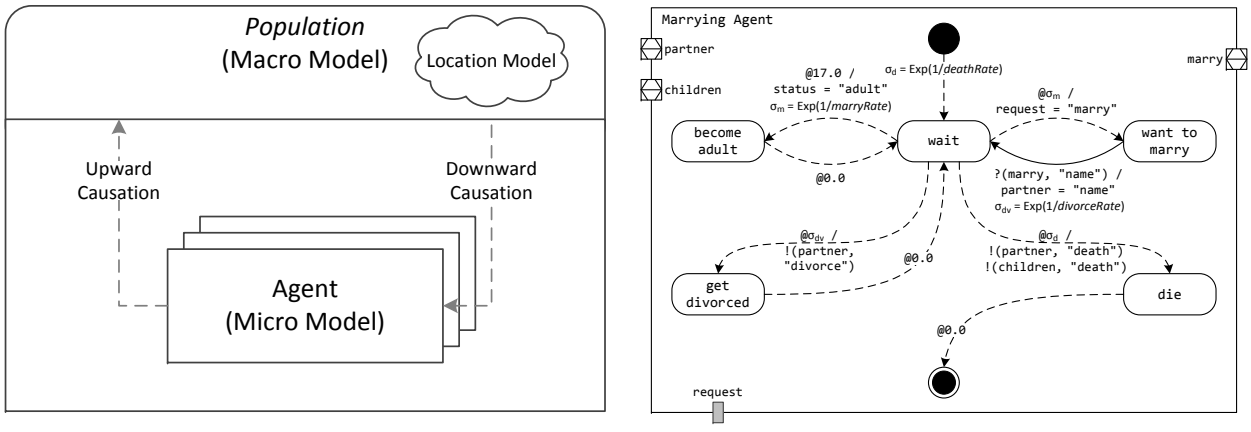

Figure 1: (Left) Composition of a ML-DEVS model realizing the *linked lives* model. The number of agents can change during simulation. (Right) The marrying behavior of an agent of the composition depicted as a state chart, where `partner`, `children`, and `marry` are ports and `request` is a public state variable.

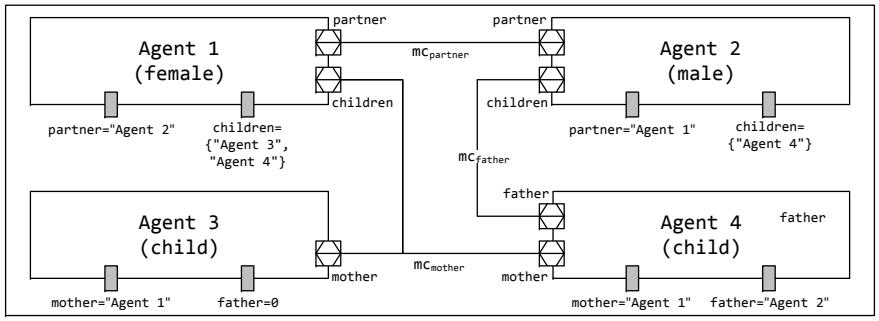

Figure 2: Coupling scheme of the ML-DEVS model. Agent 1 and 2 are partnered and have a child (Agent 3). Agent 1 has another child (Agent 4) whose father is already dead (indicated by `father = 0`).

locations (location model) are reflected in the macro model's state and are made accessible for the agents at the micro level via *value coupling* that is part of the downward causation (cf. Steiniger et al. (2012)).

As agents are represented by Micro-DEVS models that exhibit a behavior of their own, each agent can schedule its own death or, in the case that the agent is married, a divorce in terms of an internal event based on certain rates. As mentioned before, the location model is part of the macro model's state and is made accessible to all agents. In turn, each agent announces important attributes (public state), such as the name of its partner, that are required by the macro model, e.g., to link agents via multi-couplings and to maintain a consistent state. For instance, we assume that houses (locations) have a certain capacity and agents cannot move to a house that has no space left. The multi-couplings represent links between agents as shown in Figure 2. Via these couplings the respective agents can exchange events, e.g., a mother can inform her children when she dies or moves to a different location. Apart from unidirectional communication between individuals, actions involving more than one agent are controlled/performed by the macro model. For instance, individuals forming a partnership are linked by the macro model. Also adoptions are handled by the macro model. This kind of control is realized by the downward activation mechanism in ML-DEVS.

3.2 ML-Rules

ML-Rules was originally designed for multi-level modeling in computational systems biology, where different levels of organization and their interplay (up- and downward causation) are of particular interest (Maus et al. 2011). Therefore, the modeling language supports hierarchical nesting of attributed model entities (called *species*) and explicit up- and downward causation between different levels of organization. ML-Rules is a rule-based language that supports a reaction-centric view, where rules are defined globally based on nested, attributed species and serve as schemata for reactions. The underlying semantics of ML-Rules is a continuous-time Markov population model (Henzinger et al. 2011).

When we started to translate the *linked lives* model into an equivalent ML-Rules model, one drawback of the original syntax of ML-Rules became immediately apparent: the specification and handling of attributes. For each species we only define the number of its attributes, but no names via which the attributes can be accessed. As attributes are neither accessible by their indices in the internal attribute list, all attributes of a species have to be specified, i.e., either bound to temporary variables that can be used in a rule or matched against concrete values, when defining rule schemata, even if the attributes are unaffected. Moreover, the modeler has to take care that attributes are used consistently across different rule schemata. We address these issues by introducing named attributes, which allows us to pursue the strategy of "*don't care, don't write*". In contrast to the ML-DEVS model, the flexible nesting and up- and downward causation of ML-Rules can be used to resemble the organizational structure of the *linked lives* model. Thus, the ML-Rules variant of the original model comprises three main species: *Agents*, *Houses*, and *Towns*, where towns contain houses that contain agents in turn. Figure 3 shows the definition of these species including their attributes and the initial values of the attributes. The agents do not have an attribute declaring their age, instead it can

```
Town(x:0, y:0);
House(address:0, size:3, occupants:0);
Agent(birth:1950, age:0, sex:"m", mother:0, father:0, partner:0, children:0, status:"child",
    health:"healthy");
```

Figure 3: Species definitions in ML-Rules with attributes and initial values. The attributes `mother` and `father` refer to unique keys shared between the agent and its parents (see below). The attributes `partner` and `children` refer the keys shared with a potential partner and children, respectively. The attributes `status` and `health` are taken from the *linked lives* model.

be determined based on the agents' year of birth (denoted by `birth`) and the current simulation time. So similarly as in ML-DEVS, no explicit aging rules are required. Instead, when calculating the propensities of all reactions to determine which reaction has to fire next and when, we compute the actual age of the available agents. As we have no explicit notion of time in ML-Rules, accessing the simulation time is a feature that has to be provided by the simulator of ML-Rules. Figure 4 illuminates how the agents' age can be considered when specifying the rate of a rule that covers the death of agents without (living) children and partners. The rates for female and male agents are determined by the functions `getFemaleDeathRate`

```
Town()[House(occupants:oc)[Agent(birth:bi, sex:s, patner:0, children:0):a]:h]:t ->
    t[h(occupants:(oc - 1))] @ if (s == "f") then getFemaleDeathRate(age(bi)) else
    getMaleDeathRate(age(bi)) #a * #h * #t;
```

Figure 4: Rule for the death of childless agents without a partner.

and `getMaleDeathRate`, respectively, based on an agents' current age, which means the rates are time-dependent. How these rates are calculated concretely is not subject of this paper. The operators `#a`, `#h`, and `#t` return to the overall number of agents, houses, and towns that fulfill the left hand side of a concrete instance of the depicted rule, e.g., all female agents that were born in 1984 and that live in the same place. Whereas, the variables `a`, `h`, and `t` refer to a concrete agent, house, and town that are eventually affected by the reaction. These individual species can be reused and manipulated on the right hand side. If they contain other species (e.g., other agents that live in house `h`), these species remain unaffected. Relationships between different individuals (links) are central to the *linked lives* model. We establish these exclusive relationships by using the ν-operator (denoted by '`$`') of ML-Rules. For this, a unique name (key) is generated by the operator and assigned to a corresponding attribute (`mother`, `father`, `partner`, or `children`) shared by all individuals that are part of the relationship. Figure 5 shows how a global rule for the marriage of childless agents using the ν-operator could look like. The rate

```
Town()[House()[Agent(birth:bi1, partner:0, children:0, sex:"f"):a1]:h1]:t1 +
    Town()[House()[Agent(birth:bi2, partner:0, children:0, sex:"m"):a2]:h2]:t2 ->
    t1[h1[a1(partner:$p)]] + t2[h2[a2(partner:$p)]] @ if ((age(bi1) >= 17) && ((age(bi2) >=
    17))) then getMarriageRate(age(bi), age(bi2)) * #a1 * #a2 * #t1 * #t2 * #h1 * #h2 else 0;
```

Figure 5: Rule for the marriage of a male and female agent without children. After the marriage (reaction fires) the matching agents `a1` and `a2` assign a new, unique key (`$p`) to their `partner`-attributes. This shared key can later be used to identify the two agents as a couple.

of a concrete rule to fire depends on a rate determining the willingness of to marry that is calculated by the function `getMarriageRate` and on the possible combinations of women and men not yet married in the population, similar to the *mass action kinetics* in computational systems biology. Concrete, the higher the number of possible matches, the higher the probability that a marriage will take place in the near future. A further interesting aspect in our model is the moving of agents, as it leads to a change of the composition

structure and refers to another matching problem, i.e., match agents with suitable houses. For this, we specify special moving rules. Figure 6 shows one of these—a rule for childless, adult agents that still live with their parents and have no partners. However, agents can only move to houses with sufficient space.

```
Town()[House(occupants:oc1)[Agent(birth:bi, partner:0, children:0, status:st1, status:"adult␣
    at␣home"):a1]:h1]:t1 + Town()[House(size:si2, occupants:oc2)[]:h2]:t2 ->
    t1[h1(occupants:(oc1 - 1))]] + t2[h2(occupants:(oc2 + 1))[a1(status:"independent␣adult")]]
    @ if ((h1 != h2) && ((si2 - oc2) >= 1)) then getMoveRate(age(bi)) * #a1 * #h1 * #h2 * #t1 *
    #t2 else 0;
```

Figure 6: The process whereby childless adults with no partners move out of their parent's house. When the reaction fires, agent `a1` will be removed from its old house `h1` and moves into a new house `h2`, when their is space left.

In addition to the capacity we could also easily consider preferences of agents for houses with certain characteristics, which can be viewed as an illustration of *preferential attachment*. In Noble et al. (2012), the simulation is aborted in the case that no empty house are left. Therefore, the simulation has to be initialized with enough houses to get useful results. In our case, the movement would not take place, i.e., the rate would be 0. In addition, we could also think about creating new houses "on demand".

As Section 2 mentions, certain actions in the *linked lives* model can trigger others, which have to be performed consecutively. For instance, if both parents of a child are dead, the child will be adopted by another married couple and will move in with its new parents at the same time step at which the last of the child's parents died. In ML-Rules we can influence the rate with which reactions fire. By setting the rate of a rule to ∞, reactions that can be derived from this rule can immediately fire. Still, only one reaction can fire at the same time. Causal relations between different rules can be expressed by introducing special "trigger species". Figure 7 shows rules that make use of a trigger species `InformDeath` and infinite rates (the last two rules). Whenever an agent with children dies (first rule), the key that binds the agent to its children is assigned to the attribute of the trigger species, which has only one attribute. This key is unique and thus can be used to "find" the children of the late agent (last two rules). As long as one of the rules can be instantiated (i.e., there is an agent that fulfills the conditions), a reaction will be fired instantly. In this reaction, either the attribute `mother` or `father` of the affected child will be set to 0. As agents can

```
// simplified dying rule for agents with living children
Town()[House(occupants:oc)[Agent(sex:s, children:ch, birth:bi):a]:h]:t -> t[h(occupants:(oc -
    1))] + InformDeath(parent:ch) @ if (ch != 0) then getDeathRate(age(bi)) * #a * #h * #t else 0;
// update rule for mother died
InformDeath(parent:m) + Town()[House()[Agent(mother:m):a]:h]:t -> InformDeath(parent:m) +
    t[h[a(mother:0)]] @ if (m != 0) then infinity() else 0;
// update rule for father died
InformDeath(parent:f) + Town()[House()[Agent(father:f):a]:h]:t -> InformDeath(parent:f) +
    t[h[a(father:0)]] @ if (f != 0) then infinity() else 0;
```

Figure 7: Information propagation via a trigger species, here `InformDeath`, and infinite rates.

have more than one child, the rules produce another `InformDeath` species on the right hand side and respective reactions will fire until all children are informed.

3.3 Attributed Pi

Attributed π is used to describe concurrent systems composed of synchronously communicating processes. Names play a central role in the π-calculus (Milner 1999) and its variants, such as attributed π (an extension of stochastic π). Names represent communication channels on the one hand and (free and bound) variables on the other hand. In attributed π, processes are equipped with attributes that can be used

to constrain the interaction between the processes. Therefore, the call-by-value λ-calculus is used as a sequential language in which data values and constraints for concurrent interactions can be defined. Thus, attributes and interaction constraints add expressiveness to the original stochastic π calculus. Attributed π was developed in the context of computational systems biology. One of its motivation is to enable modeling spatial aspects depending on numeric attributes and dynamic compartments with various nesting structures, although only implicitly (in contrast to ML-Rules). Due to combining a concurrent process algebra, i.e., the π-calculus, and a sequential language, i.e., the λ-calculus, modeling multiple levels becomes possible, as those levels can now be encoded in the attributes and constrain the communications between the processes. Like stochastic π, attributed π has a continuous-time Markov chain semantics.

In attributed π we use attributed processes (model entities) to model the individuals of the *linked lives* model. We distinguish between children and adults, the latter can either be females or males. As attributed π models are flat by definition, we have to capture the locations by additional processes and attributes. Figure 8 shows the model of a *child* (process) having the attributes: `sex`, `father`, `mother`, `location`,

```
1 Child(sex, father, mother, location, birthday) ≜
2   @[if (sex = "f") then adult(birthday) else 0] . ν children . Female(father, mother,
        location, 0, birthday, children, 0)
3   + @[if (sex = "m") then adult(birthday) else 0] . Male(father, mother, location, 0,
        birthday, 0, 0)
4   + @death . (mother[∞]!("death") | father[∞]("death")) . 0
5   + mother[λ d.d]?(x) . Child(sex, if (!isLocation(x) && (x != "death")) then x else father,
        if (x == "death") then 0 else mother, if (isLocation(x)) then x else location,
        birthday)
6   + father[λ d.d]?(x) . Child(sex, if (x == "death") then 0 else father, mother, location,
        birthday)
7   + adoption[λ prop1, prop2 . if ((age(birthday) < 17) && (mother == 0) && (father == 0))
        then adoptRate(prop1, prop2, sex, birthday) else 0]?(x) . Child(sex, 0, x, location,
        birthday)
```

Figure 8: A model of a child. Process names start with uppercase letters, whereas names of communication channels and attributes start with lowercase letters.

and `birthday`. The behavior of a child can be described by six alternatives (lines 2 to 7 in the figure) competing with each other by stochastic race, denoted by the choice operator '+'. From the point of view of the simulation, the next events (dying, adoption, etc.) are determined by calculating the sojourn times, and what event comes first is executed. This refers to the theory of competing risks in which latent waiting times are generated and the event associated with the smallest waiting time occurs. All other options will be discarded. For instance, if the fifth alternative (line 6) is chosen and the information being sent over the channel `father` is `"death"`, then the process `Child` will continue as process `Child(sex, 0, mother, location, birthday)`, but without a father (second attribute is set to 0).

Prefixes of processes (starting with uppercase letters) mean that a process is either listening on a channel (indicted by '?') or sending information via a channel (indicated by '!'). For instance, the prefix `adoption[λ prob1, prob 2 . if ...]?(x).Child(...)` means that a child can listen on the public adoption channel (called `adoption`) and the information sent via this channel is the name of a (new) private channel, i.e., `x`. The actual rate of an adoption to happen can be determined as follows: If the child has still one parent or is older than 17 years, the rate will be 0, which means no adoption will take place. If an adoption should take place, its rate is determined by the function `adoptRate` based on: (a) the attributes of the potential mother (her age and the number of her children (see also Figure 10)) and (b) the attributes of the child to adopt (sex and age of the child). The former information is received by the child via the adoption channel and sent by the female willing to adopt a child (Figure 10). The rate determines how long a child is waiting for being adopted and is returned by an anonymous λ-function, which takes `prop1` and `prop2` (denoting the number of children and the birthday of the potential mother) as inputs and applies itself to these properties and the attributes of the child to be adopted. Please note,

this is an extension of the *linked lives* model we included to clarify how the λ expression works. If an adoption eventually takes place, the adopted child will have a new mother, which is specified by assigning a new private channel name provided by the adoptive mother (sent over the public adoption channel) to the respective attribute of the child, i.e., `mother`. This private channel is exclusively used for the communication between a mother and all her children. As soon as a girl becomes an adult (Figure 8, line 2), this private channel name (denoted by `children`) is generated by the ν-operator.

```
1 Female(father, mother, location, partner, birthday, children, number) ≜
2 ...
3   + adoption[number, birthday]!(children).children? . Female(father, mother, location,
        partner, birthday, children, number + 1) .
4 ...
```

Figure 9: Female adopting a child by sending the child its key (denoted by `children`)

In attributed π some aspects of the *linked lives* model are cumbersome to model. For instance, without further effort it is not possible to inform all children about a moving event of their mother simultaneously, as only two processes can communicate at once. We address this problem by using special processes (similar to the trigger species in ML-Rules), which are responsible to communicate over a channel certain information to a number of processes, such as `InformProcesses(number, channel, info)` in Figure 10. Figure 10 illuminates the usage of such a special process, i.e., `InformProcesses(number, channel, info)`. The movement of a female depends on her properties, her family, and the currently available locations. The latter are stored in the attribute `location` of the process `Location`.

```
1  Female(father, mother, location, partner, birthday, children, number) ≜
2    + ν askForMov .
3        move[number, children, location]!(number, children, location, askForMov) .
4        askForMov[λ x.x ]?(newLocation) . (InformProcesses(1, partner, newLocation) |
5            InformProcesses(number, children, newlocation) | Female(father, mother,
                 newLocation, partner, birthday, children, number))
6  ...
7  Location(locations) ≜
8  ...
9    + move[λ prop1, prop2, location . moveRate(prop1, prop2, location, locations)]?(prop1,
         prop2, prop3, x) .
10         x[∞]!(newLocation(prop1, prop2, prop3, locations)) .
               Location(updateLocations(locations, prop1, prop2, prop3))
11 ...
12 InformProcesses(number, channel, info) ≜
13     channel[if number > 0 then ∞ else 0]!(info) . InformChildren(number - 1, channel, info).
```

Figure 10: Female moving and informing her partner and her children.

Note that for demographic models it is essential to access the simulation time, e.g., in Figure 8 the function `adult(birthday)` determines the time at which a child turns into an adult by taking `birthday` and the current simulation time into account. As attributed π currently does not provide this functionality, its applicability for demographic models is generally rather limited.

3.4 Discussion

All of the considered modeling approaches describe discrete event systems. Thus, in all approaches probabilities of the time-stepped model have to be translated into sojourn times. Due to the widespread availability of discrete time-stepped simulation tools or the simplicity to develop discrete time-stepped simulators from scratch, currently most demographic micro models are discrete stepwise models, although it is frequently argued that continuous-time microsimulation mirrors life course developments more closely

(Willekens 2005). In addition, discrete event simulation is often more efficient. All of the above approaches support variable structure models, i.e., changes of composition, behavior, and interaction patterns. However, they do so differently: Whereas ML-DEVS constrains changes to the subordinate level in the composition hierarchy, structure changes can occur globally in ML-Rules. That way compartments can merge, entities can migrate from one compartment to another, etc. In attributed π, new processes and communication channels between processes can be generated in arbitrary numbers.

All modeling languages/approaches considered here support rather different views on the system of interest. Although, the reactive systems metaphor of the modular, hierarchical modeling formalism ML-DEVS (Steiniger et al. 2012) appears close to the traditional agent metaphor, the resulting model is anything but compact. The reaction-centric ML-Rules model emphasizes what can happen to and between the individuals. This depends very much on the respective state and context of individuals. Thus in comparison to biochemical models, where species often react with each other independently of the comportment in which they are located and independently of further attributes, the reaction rules in our demographic model tend to be more complex than reaction rules typically encountered in cell biology. The third approach, i.e., attributed π, describes the demographic system as a set of communicating concurrent processes. It provides the most compact description, as it focuses on individuals as concurrent processes communicating with each other. However, this compactness is bought by the combination of a concurrent and a sequential language and is, thus, likely to require more effort for a domain expert to understand.

In (agent-based) demographic models, the aging process of agents plays an crucial role. Thus the simulation time needs to be accessed. Typically, modeling languages developed for systems biology, i.e., for modeling continuous-time Markov populations, do not support this demand, which makes their applicability for modeling demographic systems problematic. Also the fact that individuals can belong to several overlapping groups, such as kin networks or communities, is rather specific for demography. In DEVS and its variants, the "part-of"-relation is typically exclusive (a few exceptions such as Dalle et al. (2008) exist). Similarly, in ML-Rules the nesting structure is interpreted as a spatially being nested, which means an entity cannot be at two locations at the same time. Thus, the membership in different groups cannot be realized by hierarchical structures offered in both, ML-DEVS and ML-Rules, but has to be modeled differently. Instead, groups need to share a common communication platform, such as the private channels in attributed π. In ML-DEVS this is realized by the multi-couplings (in basic DEVS this would have been more challenging) and in ML-Rules by species sharing attributes. However, as the numbers of attributes of species are fixed, this modeling is more restrictive than the one in attributed π and ML-DEVS. In sum, we find that all the presented modeling approaches allow specifying the *linked lives* model, but are cumbersome and often far from being intuitive. This is mostly due to the fact that they are tailored to describe biological systems rather than demographic systems.

4 THE ROLE OF LANGUAGES FOR VALIDATING DEMOGRAPHIC MODELS

The tendency (and also argued need) in demography to shift toward models that are enriched by hypotheses about causal mechanisms (which may drive an individual's decision, e.g., to marry or migrate) requires a different kind of approach for estimating parameters and generally for validation (Klügl 2008, Louie and Carley 2008). This approach relies on experimenting with the model at hand. Experimental parameter estimation methods follow a simulation-based optimization scheme (Fu 2002), where the model is given and only the values of selected parameters need to be determined. Therefore, experiments are run while the parameters are systematically varied until the value of a target function, e.g., the distance between observed and simulated behavior, reaches a certain threshold. To search the parameter space efficiently, especially if the space is large, a variety of methods exists (Sanchez and Wan 2012). This variety as well as the nature of the experimentation process make additional support desirable (Perrone et al. 2012). This can be in terms of helping to select methods for individual steps of this process, such as execution algorithms to increase efficiency (Helms et al. 2013), or referring to the process of experimenting and its documentation (Rybacki et al. 2011). With respect to the latter, we will focus on the simulation experiment specification

```
import sessl._
import sessl.james._

new Experiment with Observation with ParallelExecution with DataSink {
  model = "file-sr:/./linkedlifes.mlr"
  replications = 10; stopTime = 50
   scan(
    "divorceRate" <~ range(0.01, 0.01, 0.10))
  observe("Town/House/Agent(children:0)")
  observeAt(range(0, 0.1, 50))
  dataSink = MySQLDataSink(schema = "test2")*
  rng = MersenneTwister()
  simulator = DirectMethod()
}
```

Figure 11: A SESSL experiment using JAMES II. (Scala keywords are shown in blue.)

language SESSL (Simulation Experiment Specification via a Scala Layer) (Ewald and Uhrmacher 2014). The motivation of SESSL is to facilitate the generation, documentation, and reproduction of experiments. SESSL is an embedded domain-specific language, which is a language that is tailored for a specific application domain and implemented by using constructs of the host language the domain-specific language is embedded into (in this case Scala). Figure 11 shows how a simple parameter scanning experiment, like those described by Noble et al. (2012), could look like in SESSL.

5 CONCLUSION

Typically, languages (and modeling approaches) are designed to support domain experts in understanding, validating, changing, and developing models and experiments. Based on the three modeling languages and approaches ML-Rules, attributed π, and ML-DEVS, we demonstrated essential features of multi-level modeling in demography. Among the different perspectives on dynamic, continuous-time, discrete event systems, i.e., reactive systems, reaction-centric, and concurrent processes, the latter provides the most succinct definition of continuous-time, demographic micro models. However, syntax matters and the particular combination of a concurrent and sequential language, i.e., the π- and λ-calculus, which makes this approach powerful, comes with a price: the learning curve for a typical domain expert is expected to be steep. The question is whether a domain-specific language can be developed, which allows a similarly compact and succinct definition of models based on idioms and abstractions at the level of the demographic domain. Essentials of this domain, i.e., accessing simulation time to take the aging process of individuals into account and the membership of those in multiple groups, have been emphasized by translating the *linked lives* model into the different approaches. The study of other models, e.g., the *Wedding Ring* and the *Wedding Doughnut*, also revealed that groups and the interaction between the members of those groups are dynamic. Thus, variable structure modeling is crucial for agent-based demographic models.

Languages do not only play an important role for modeling, although here the most obvious, but also for specifying experiments with demographic models, as our short excursion into the simulation experiment specification language SESSL showed. Generally, specialized languages allow expressing solutions at the level of abstraction of the problem domain, whether this is modeling migration processes of European countries or specifying parameter scanning experiments with these kinds of models.

ACKNOWLEDGMENTS

Special thanks to Stefan Rybacki, who helped us with streamlining the ML-Rules model.

REFERENCES

Billari, F., A. Prskawetz, B. Aparicio Diaz, and T. Fent. 2007. "The "Wedding-Ring": An agent-based marriage model based on social interaction". *Demographic Research* 17:59–82.

Bohk, C., R. Ewald, and A. M. Uhrmacher. 2009. "Probabilistic Population Projection with James II". In *Proceedings of the 2009 Winter Simulation Conference*, edited by A. Dunkin, R. G. Ingalls, E. Yücesan, M. D. Rossetti, R. Hill, and B. Johansson, 2008–2019. Piscataway, New Jersey: IEEE.

Cardelli, L., and P. Gardner. 2010. "Processes in Space". In *Proc. of Conf. on Computability in Europe*.

Dalle, O., B. P. Zeigler, and G. A. Wainer. 2008. "Extending DEVS to Support Multiple Occurrence in Component-Based Simulation". In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. Hill, Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler, 933–941. Piscataway, New Jersey: IEEE.

Ewald, R., and A. M. Uhrmacher. 2014. "SESSL: A Domain-Specific Language for Simulation Experiments". *ACM Transactions on Modeling and Computer Simulation* 24 (2).

Ewert, U. C., M. Röhl, and A. M. Uhrmacher. 2007. "Hunger and market dynamics in pre-modern urban communities: insights into the effects of market intervention from a multi-agent model". *Historical Social Research* 32 (4): 122–150.

Faeder, J., M. Blinov, and W. Hlavacek. 2009. "Rule-Based Modeling of Biochemical Systems with BioNetGen". *Systems Biology, Methods in Molecular Biology* 500 (2): 1–55.

Fu, M. C. 2002. "Feature article: Optimization for simulation: Theory vs. practice". *INFORMS Journal on Computing* 14 (3): 192–215.

Gilbert, N., and K. Troitzsch. 2008. *Simulation for the Social Scientist*, Chapter Multilevel simulation models, 100–129. Open University Press.

Helms, T., R. Ewald, S. Rybacki, and A. M. Uhrmacher. 2013. "A Generic Adaptive Simulation Algorithm for Component-based Simulation Systems". In *Proceedings of the 2013 ACM SIGSIM Conf. on Principles of Advanced Discrete Simulation*, 11–22.

Henzinger, T., B. Joostmann, and V. Wolf. 2011. "Formalism for specifying markovian population models". *International Journal of Foundations of Computer Science* 22 (4).

Jager, W., and M. A. Janssen. 2003. "Diffusion processes in demographic transitions: a prospect on using multi agent simulation to explore the role of cognitive strategies and social interactions". In *Agent-Based Computational Demography*. Physica-Verlag.

John, M., C. Lhoussaine, J. Niehren, and A. M. Uhrmacher. 2010. "The attributed pi-calculus with priorities". In *Transactions on Computational Systems Biology XII*, Volume 5945, 13–76. Springer-Verlag.

Kim, J.-H., and T. G. Kim. 2001. "DEVS-Based Framework for Modeling/Simulation of Mobile Agent Systems". *SIMULATION* 76 (6): 345–357.

Klügl, F. 2008. "A validation methodology for agent-based simulations". In *Proceedings of the 2008 ACM Symposium on Applied Computing*, 39–43: ACM.

Krivine, J., R. Milner, and A. Troina. 2008. "Stochastic Bigraphs". *Electronic Notes in Theoretical Computer Science* 218:73–96.

Louie, M. A., and K. M. Carley. 2008. "Balancing the Criticisms: Validating Multi-Agent Models of Social Systems". *Simulation Modelling Practice and Theory* 16 (2): 242–256.

Maus, C., S. Rybacki, and A. M. Uhrmacher. 2011. "Rule-based multi-level modeling of cell biological systems". *BMC Systems Biology* 5:166.

Milner, R. 1999, June. *Communicating and Mobile Systems: The Pi-Calculus*. 1st ed. Cambridge, UK: Cambridge University Press.

Nikolai, C., and G. Madey. 2009. "Tools of the Trade: A Survey of Various Agent Based Modeling Platforms". *Journal of Artificial Societies and Social Simulation* 12 (2): 2.

Noble, J., E. Silverman, J. Bijak, S. Rossiter, M. Evandrou, S. Bullock, A. Vlachantoni, and J. Falkingham. 2012. "Linked Lives: The Utility of an Agent-Based Approach to Modeling Partnership and Household Formation in the Context of Social Care". In *Proceedings of the 2012 Winter Simulation Conference*,

edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, Article No. 93. Piscataway, New Jersey: IEEE.

Perrone, L. F., C. S. Main, and B. C. Ward. 2012. "Safe: Simulation Automation Framework for Experiments". In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, Article No. 396. Piscataway, New Jersey: IEEE.

Rybacki, S., J. Himmelspach, F. Haack, and A. M. Uhrmacher. 2011. "WorMS - A Framework to Support Workflows in M&S". In *Proceedings of the 2011 Winter Simulation Conference*, edited by S. Jain, R. R. Creasey, J. Himmelspach, K. P. White, and M. Fu, 716–727. Piscataway, New Jersey: IEEE.

Sanchez, S. M., and H. Wan. 2012. "Work Smarter, not Harder: a Tutorial on Designing and Conducting Simulation Experiments". In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, Article No. 170. Piscataway, New Jersey: IEEE.

Silverman, E., J. Bijak, J. Noble, V. Cao, and J. Hilton. 2012. "Semi-artificial models of populations: connecting demography with agent-based modelling". In *4th World Congress on Social Simulation*.

Steiniger, A., F. Krüger, and A. M. Uhrmacher. 2012. "Modeling Agents and their Environment in Multi-Level-DEVS". In *Proceedings of the 2012 Winter Simulation Conference*, edited by C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, Article No. 233. Piscataway, New Jersey: IEEE.

van der Gaag, N. 2009. "MicMac Final Report 2005 – 2009". Technical report, NIDI, The Hague. Final Project Report to the European Comission.

Willekens, F. 2005. "Biographic forecasting: bridging the micro-macro gap in population forecasting". *New Zealand Population Review of Economics and Statistics* 31 (1): 77–124.

Willekens, F. 2009. "Continuous-time microsimulation in longitudinal analysis". In *New Frontiers in Microsimulation Modelling*, 353–376. Ashgate.

Zinn, S. 2011. *A Continuous-Time Microsimulation and First Steps Towards a Multi-Level Approach in Demography*. Ph. D. thesis, University of Rostock, Rostock, Germany.

Zinn, S., J. Gampe, J. Himmelspach, and A. M. Uhrmacher. 2010. "A DEVS model for demographic microsimulation". In *Proceedings of the 2010 Spring Simulation Multiconference*. New York, New York, USA: Society for Modeling & Simulation International.

AUTHOR BIOGRAPHIES

ALEXANDER STEINIGER is a Ph.D. student in the modeling and simulation group at the University of Rostock. His email address is alexander.steiniger2@uni-rostock.

SABINE ZINN is a researcher at the Leibniz Institute for Educational Trajectories in Bamberg and at the Max Planck Institute for Demographic Research (MPIDR) in Rostock. She has a diploma in Business Mathematics and a Ph.D in Computer Science. Her email address is zinn@demogr.mpg.de

JUTTA GAMPE is the Head of the Laboratory of Statistical Demography of the MPIDR for Demographic Research. She received her Ph.D. in Statistics. Her research interest are statistical demography and related topics. Her email address is gampe@demogr.mpg.de

FRANS WILLEKENS is former director and Honorary Fellow of the Netherlands Interdisciplinary Demographic Institute (NIDI), The Hague and Emeritus Professor of the University of Groningen. He is also Senior Research Scientist at the MPIDR. His email address is willekens@demogr.mpg.de

ADELINDE M. UHRMACHER is professor at the Institute of Computer Science, University of Rostock and head of the modeling and simulation group. Her email address is adelinde.uhrmacher@uni-rostock.de.