Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

MIC-CORE: A TOOL FOR MICROSIMULATION

Sabine Zinn
Jutta Gampe

Max Planck Institute for Demographic Research
18057 Rostock, Germany

Jan Himmelspach
Adelinde M. Uhrmacher

University of Rostock
18059 Rostock, Germany

ABSTRACT

Microsimulation is an increasingly popular tool in the social sciences. Individual behavior is described by a (commonly stochastic) model and subsequently simulated to study outcomes on the aggregate level. Demographic projections are a prominent area of application. Despite numerous available tools often new software is designed and implemented for specific applications. In this paper we describe how a modeling and simulation framework, JAMES II, was used to create a specialized tool for population projections, the MIC-CORE. Reusing validated and well-tested modeling and simulation functionality significantly reduced development time while keeping performance levels high. We document how the MIC-CORE was built as plug-ins to JAMES II and illustrate the performance of the resulting tool. We demonstrate how the concept of a modeling and simulation framework enabled successful software reuse of available functionality and briefly report of future work.

1 INTRODUCTION

Whenever new software is about to be developed it is a good idea to search the market for existing solutions: the development of a new software is always a time consuming and error prone task. But you'll have to find a software solution which can (1) cope with the domain, (2) can be adapted to the needs of the user group, (3) which is trustworthy, and (4) where you can take care on bug-fixing your own, or where you have sufficient support from the developers. The number of existing solutions in the market to be checked is pretty high in the field of modeling and simulation (M&S) (http://eprints.agentlink.org/view/type/software.html, accessed April 2009, lists 128 multi agent software items; http://www.sce.carleton.ca/faculty/wainer/standard/tools.htm, accessed April 2009, lists 15 tools for DEVS based M&S), and thus, although you might be able to exclude lots of the potential candidates by some "simple filtering" mechanisms as, "only open source", a great number of solutions may remain. These solutions exist in a variety of application domains (e.g., multi agent simulation, micro simulation, network simulation), or they are designed to be more general purpose (i.e., for discrete - event, hybrid or continuous simulation), and it is hard to get a complete overview of all existing solutions. And it is even harder to make a fair comparison of these. So people might end up creating their own tool although they wanted to reuse. Besides reusing in the large (thus complete tools here), one could try to build the new application based on a library or a framework (reuse in the small) - this should significantly decrease the overall amount of work to be done, and should help creating well-tested applications, as long as the libraries or frameworks used are well-tested. But up to which degree can a framework support the creation of a specialized M&S tool? In the context of the project 'MicMac (www.micmac-projections.org, accessed April 2009) - Bridging the micro-macro gap in population forecasting' a microsimulation tool, the MIC-CORE, had to be developed. The aim of the MicMac-project was "to develop a methodology and the associated software that offers a bridge between aggregate projections of cohorts (Mac) and projections of the life courses of individual cohort members (Mic)." (NIDI 2006). Both Mac and Mic have been implemented in one software tool. The Mac part has been implemented using a standard methodology for demographic projections called cohort component model (van Imhoff 1990). The MIC-CORE comprises the Mic part of the project. The main purposes of its development have been the design of a software that can easily be used without assuming background knowledge in M&S and the employment of computationally efficient simulation algorithms to handle large-scale problems. The tool (MIC-CORE) described here has been created based on the M&S framework JAMES II. The most important idea while developing JAMES II was to create a framework which allows to create more specialized applications by extending it. So far this had only been done by some of the core developers. However, the question arose how this would work for someone

external - which parts of the frameworks are reusable, and what are the benefits of this for a resulting tool? In the following we'll first give a short introduction into microsimulation, we motivate the creation of a new tool, and thereafter show how the tool has been realized, outlining the actual reuse. The tools usability is shown by a simple model and its validation. Finally we conclude with summing up our experiences, and give an outlook of how the resulting tool, MIC-CORE, can be extended in the future.

2 BACKGROUND

2.1 Microsimulation for Population Projections

In the social sciences, including demography, microsimulation denotes approaches for which the behavior of the basic micro level unit (typically an individual or a household) is modeled and simulated rather than the aggregate behavior of a system, i.e., a population, society or economy. Microsimulation goes back to the work of Orcutt (Orcutt 1957, Orcutt, Greenberger, Korbel, and Rivlin 1961) and is an obvious method to produce population projections (Imhoff and Post 1998, Wolf 2001). Individuals make up the population and their distribution over age-classes, marital status, labour force participation, etc. ultimately determines the population structure, which is of crucial importance for policy decisions and planning purposes. Hence we may simulate future changes in individual demographic behavior to learn about prospective changes in population size and structure.

The central unit of a demographic microsimulation is an individual's life course, which is characterized by a sequence of demographic events such as birth, marriage, childbirth, divorce, retirement and finally death. The length of spells between these events gives, together with the individual's birth date, the ages at which these events are experienced. Despite considerable regularity of demographic behavior the order and age-specific incidence of demographic events varies between individuals and cannot be fully explained by observable characteristics. Therefore individual life courses are appropriately modelled by stochastic processes (Bartholomew 1973, Mayer and Tuma 1990). The propensity for experiencing certain demographic events is usually age-dependent but also varies with calendar time (e.g., decreasing mortality or increasing divorce rates), and the corresponding current age- and time-specific incidence rates can be estimated from vital statistics and social surveys. Assumptions about future rates then define the projection scenarios.

A microsimulation model for population projection therefore consists of a virtual population and a stochastic model of individual behavior (Willekens 2007). The virtual population consists of individuals characterized by attributes like sex, current marital status, educational attainment, children ever born, etc. and their distribution usually resembles current population structure. The model population evolves over time because based on the stochastic model individuals experience certain demographic events over their life time. Additionally to the individuals in the starting population, new members may enter by birth and, if a so called open population model is used, by immigration. Individuals exit the population by death, and in open models by emigration.

In general one can distinguish between static and dynamic microsimulation models (Gilbert and Troitzsch 2008, Brown and Harding 2002). In static models changes in the population structure are realized by reweighting individual data records for the distinct years in the simulation period. In dynamic models aging the micro units individually leads to a time-varying population structure. Hence, demographic microsimulation as described above employs a dynamic model. In short, a "microsimulation consists of drawing a sample of realizations of a pre-specified stochastic process" (Wolf 2001). Thereby, the pre-defined stochastic process is part of the simulation model and the sampling procedure is part of the execution of the model. Before a model execution can be performed, the stochastic process and related parameter values have to be specified. Several alternative model variants are available. First, the time scale in such a model can either be discrete or continuous (Galler 1997). Also we have to decide on the dependence structure of events (within one individual) across the life course. Mostly the processes applied in recent microsimulation models are Markovian or semi-Markovian (O'Donoghue 1999, Wolf 2001). In this case, together with the initial population, a set of transition rates or transition probabilities, respectively, specifies the microsimulation model. The required parameters of the stochastic process must be obtained by statistical estimation (for past and current rates) and other means (for future rates), such as experts' assessments or extrapolation. To make a stochastic process a realistic model of individual behavior we usually have to choose a rather detailed model structure, which leads to considerable requirements concerning good micro data. This is one of the major challenges to the microsimulation methodology. Furthermore, as demographic applications usually intend to mimic national populations, efficient and powerful computer technology is necessary to execute large scale and complex population projections by microsimulation.

2.2 Stochastic Model for Individual Life Courses

As outlined above both age and calendar time will have to enter a realistic demographic microsimulation model, and both can be treated as discrete (usually in units of years) or continuous. As many of the vital statistics data are collected annually and grouped by single years of age a discrete approach seems natural. However, dynamic microsimulation models where calendar time is treated as discrete variable inherit some problems related to inefficient model execution and logical reasoning. Such models are executed stepwise. This means time advances in discrete steps, mostly in years. At each time step an update of each attribute of each individual of the virtual population has to be performed. This makes the simulation processing computationally expensive and inefficient (Satyabudhi and Onggo 2008). Furthermore, as only discrete time intervals are considered the model builder has to face the problem of multiple events within these time intervals (Willekens 2007, O'Donoghue 1999).

Dynamic microsimulation models where calendar time is modeled as a continuous variable offer possibilities to overcome these problems (Willekens 2007). However, the modeler has to face some problems here as well. These are mainly related to somewhat more complex parameter estimation problems (Galler 1997), and to the alignment of the microsimulation output to benchmark macro estimates like to external projections of aggregated or group variables (Harding 2007). An extensive discussion on how these problems can be handled is given in (Willekens, de Beer, and van der Gaag 2007a) and (van der Gaag, de Beer, and Willekens 2005). In our view, a dynamic microsimulation model with continuous time is the most appropriate model for population forecasting.

2.2.1 Continuous-Time Multi-State Model

The specific dynamic microsimulation model we consider is a continuous-time multi-state model. This model class is generally used for describing individual life courses (Andersen and Keiding 2002). A multi-state model is a stochastic process that at any point in time occupies one out of a set of discrete states (Hougaard 1999). These states summarize the demographically relevant categories an individual can belong to. Any demographic event changes the individual's state. For example, an unmarried women with no children, who holds a university degree and lives with her partner, changes her state by giving birth to her first child. Likewise other events such as marriage or end of partnership would imply a change of state. In this terminology states characterize the individual's relevant demographic attributes. The possible states hence depend on the specific application, but commonly they will at least comprise the elementary demographic characteristics of sex and marital status. As an individual's state usually is a combined characteristic, given by the combination of several attributes, we define so called state variables. All the unique combinations of values of these state variables define the state space, which is the set of all possible states of a multi-state model. The state space is denoted by Σ. To give an example, we describe the state space considered in the MicMac-project (Willekens, de Beer, and van der Gaag 2007b). The state variables and their corresponding values, which are given in parentheses, are: Sex (male, female), Marital Status (never married, married, divorced, widowed), Living arrangement (child in parental home, living without partner, living with partner, living with other person(s), living in institution), Disability status (disabled, non-disabled), Smoking (never, ever, current), and Children ever born $(0,1,2,\ldots)$. A state in the state space is given by a combination of values, one for each state variable. For example, [female, married, living with partner, non-disabled, current, 0] would be one potential state in this state space. Whenever one of the state variables changes its values the process changes its state, i.e., the individual experiences an event (also called state transition). An individual's life course is defined as the sequence of experienced events, and the length of spells between two events. To facilitate notation, in the following we simply enumerate the states and hence denote the state space by $\Sigma = \{1,2,\cdots,I\}$, where I is the number of all possible states. Whether and when an individual experiences an event is embodied in the transition rates of the multi-state model. Generally, we only use models for which the transition rates depend on (Gampe and Zinn 2007) the current state (and not on the previously occupied states), the age at which the individual entered the current state, and the calendar time. In principle these transition rates could additionally also depend on the time already spent in the current state but this variant is not considered in the current model. As we consider continuous multi-state models, individual life histories evolve along two continuous time scales (Wolf 1986), namely age and calendar time. Once the state space and the age- and time-specific transition rates are given we are able to simulate life courses of individuals following this multi-state model. In an open population model individuals enter the population in an initial state $i_0 \in \Sigma$ at age x_0 and calendar time t_0. Three options exist to become part of the virtual population: (1) by birth, (2) by immigration, (3) by being part of the initial population. For each individual in the initial population the age and the state occupied at the starting time of the simulation is assigned. (Usually, the initial population mimics the current population structure). Births are the result of fertility related state transitions of females in the virtual population. Sex of newborns is assigned according to a defined sex-ratio (commonly 50:50). The remaining state variables are set to the obvious values (never married, child

in parental home, ...). Finally, immigration is executed according to given numbers of immigrants that are categorized according to immigration year, age at immigration and state occupied at immigration. Immigration patterns are externally given.

Formally, a life course is conceptualized as a trajectory of a semi-Markov process $Z_t = J_{N(t)},\ t \geq 0$. The associated two-dimensional stochastic process $(J_n, T_n)_{n\in\mathbb{N}_0}$ describes the evolution in time of an individual life course. Directly linked to this process is the non-homogeneous Markov Chain $(J_n)_{n\in\mathbb{N}_0}$ that maps the states visited by an individual. The subprocess $(T_n)_{n\in\mathbb{N}_0}$ gives the sequence of consecutive transition times of the process $(J_n, T_n)_{n\in\mathbb{N}_0}$. The random number $S_n = T_{n+1} - T_n$ is the waiting time in the state J_n. The hazard rate (or transition rate) that describes the transition to state j, given that the individual occupies state i at calendar time t and age x is defined as

$$\lambda_{ij}(\tau \mid t,x) = \lim_{h\downarrow 0} \frac{1}{h}\, P\big[J_{n+1} = j, S_n \in (\tau, \tau+h] \mid J_n = i, C(T_n) = t, A(T_n) = x, S_n > \tau\big].$$

Here we include $A(T_n)$ and $C(T_n)$ to express the age- and time-dependence of the process. The function $A(T_n)$ maps the (biological) age at T_n and the function $C(T_n)$ the calendar time at T_n (Wolf 1986). The hazard rates of a semi-Markov process are its key quantities. Once they are known one can compute the distribution functions of the sojourn times.

In the subsequent S_{ij} denotes the random waiting time in state i before experiencing a transition to state j. To each rate $\lambda_{ij}(\tau \mid t,x)$ corresponds a random waiting time S_{ij}. The distribution function of S_{ij} is denoted by $F(s_{ij} \mid t,x)$, which is dependent on calendar time t and age x. It is

$$F(s_{ij} \mid x,t) = P(S_{ij} \leq s_{ij} \mid x,t) = 1 - \exp\{-\Lambda_{ij}(s_{ij} \mid x,t)\}, \tag{1}$$

where

$$\Lambda_{ij}(s_{ij} \mid x,t) = \int_0^{s_{ij}} \lambda_{ij}(u \mid x,t)\, du.$$

$\Lambda_{ij}(s_{ij} \mid x,t)$ is the so called integrated (or cumulated) hazard. As the distribution of the sojourn times S_{ij} depends on the shape of the transition rate $\lambda_{ij}(\tau \mid t,x)$, which mainly is empirically estimated from data, we usually cannot pick one of the commonly used parametric waiting time distributions. Rather we have to be able to simulate waiting times from arbitrary distributions by exploiting relationship (1).

So far an individual life course is defined as a sequence of events. Alternatively, a life course can also be described as the sequence of waiting times between events. Adopting this approach, we achieve individual life courses by using random waiting times (Gampe, Zinn, Willekens, and van den Gaag 2007). In order to clarify this procedure, we consider only one individual of the virtual population. Hereby we make the assumption that the individual just entered state i at age x and time t. If i is an absorbing state, then the individual will never leave i. In life course analysis examples for absorbent states are death and emigration. Otherwise, if i is a transient state, then the individual will move to one of the remaining $I-1$ states of Σ. The destination state within these $I-1$ states depends on "simulated" waiting times. More precisely: We compute waiting times s_{ij} for all possible destination states by using the well-known inversion theorem (Rubinstein and Kroese 2008): Replacing $F(s_{ij} \mid t,x)$ in formula (1) with a standard uniformly distributed random number $u \sim U_{[0,1]}$ leads to a random time s_{ij} from the correct distribution:

$$s_{ij} = \Lambda_{ij}(s_{ij} \mid x,t)^{-1}\left[-\ln(1-u)\right], \qquad j \neq i, j \in \Sigma. \tag{2}$$

Among all simulated waiting times, we pick the shortest waiting time $s_{ij^\star}$ to state $j^\star$. Then the individual under consideration experiences a transition to $j^\star$ at age $x + s_{ij^\star}$ and calendar time $t + s_{ij^\star}$. Such an approach, where destination states compete, is called 'competing risk' setting (Klein and Moeschberger 2003). This computation of 'shortest' waiting times is repeated for each individual of the virtual population along his/ her complete life course until the pre-set simulation end time is reached.

Equation (2) implies that we have to be able to invert the integrated hazard $\Lambda_{ij}(s_{ij} \mid x,t)$. For some basic distributions the inverse of the integrated hazard can be computed analytically, for others this can only be done numerically. A suitable approximation to $\Lambda_{ij}(s_{ij} \mid x,t)^{-1}$ is the application of piecewise linear cumulative hazards (Gampe and Zinn 2007). For the estimation of hazard rates from data and related hypothesis tests we refer to the extensive literature, see e.g. (Klein and Moeschberger 2003, Kalbfleisch and Prentice 1980).

3 MICROSIMULATION TOOLS AND M&S FRAMEWORKS: A SURVEY

3.1 MicMac Microsimulation (Requirements)

We had to find out whether we can reuse any existing tool or whether we have to implement a new tool for our microsimulation. When designing the MIC-CORE the following principles concerning the design of the software have been considered (Gampe, Zinn, Willekens, and van den Gaag 2007, Willekens 2005).

- The MIC-CORE should be freely available. Furthermore, the user should not be forced to use commercial products for running the software, for preparing required input, and for analyzing software output.
- The software should be easy to use which implies a simple graphical user interface (GUI) and the possibility to run the MIC-CORE on common desktop computers.
- In order to ensure transparency the software should be open source software.
- A linkage to a specific operating system should be avoided.
- The software should be computationally efficient in handling large-scale models.
- Up-to-date simulation technology should be used and such technology should be reused to a maximum degree wherever possible.

This list of requirements was used to check whether any microsimulation tool or a tool from other M&S domains available fits our needs.

3.2 Microsimulation Tools

Before we started to realize a new tool the structure of a number of existing dynamic microsimulation models have been studied in order to gain insights concerning current developments of microsimulations, the reusability of microsimulation tools (or parts of them) for the purposes of the MicMac microsimulation, and features and drawbacks of existing tools. As a first results we found that most dynamic microsimulation models treat calendar time as a discrete variable. We already mentioned that this approach may imply problems related to inefficient model execution and to the handling of multiple events within time intervals (cp. section 2.1). Models for year-wise execution are implemented, e.g., in UMDBS (Sauerbier 2002), DESTINIE (Duee 2005), DYNASIM/2/3 (Favreault and Smith 2004), DYNACAN (Morrison and Dussault 2000).

Some dynamic microsimulation tools incorporate a mixture of the discrete-time and the continuous-time approach. For example in SOCSIM (Hammel, Mason, and Wachter 1990) and in the demographic module of DYNAMOD-2 (King, Baekgaard, and Robinson 1999) an individual attribute is updated in that month when the related generated waiting time is expired. Otherwise, the Swedish microsimulation MICROHUS (Klevmarken and Olovsson 1996) uses continuous time duration models to simulate fertility and the formation and dissolution of consensual unions. Remaining processes are set up in discrete-time. To the knowledge of the authors, only two microsimulation tools exist where a pure continuous-time approach has been implemented: (1) PENSIM developed by the Policy Simulation Group and the U.S. Departement of Labour, and (2) LIFEPATH and variants of LIFEPATH developed by Statistics Canada. PENSIM is a pure cohort model that simulates life histories for analyzing the average lifetime coverage and adequacy issue connected to employer-sponsored pension plans in the United States (Holmer, Janney, and Cohen 2009). The stochastic simulator of PENSIM has been written in a single purpose C++ program. LIFEPATH (Statistics Canada 2004) has been set up within a general-purpose microsimulation environment "ModGen" which is a shortcut for Model Generator (Statistics Canada 2009). The environment is equipped with its own generic simulation language that is a subset of the C++ programming language (Spielauer 2006). ModGen can be used to establish microsimulation models that are variants of LIFEPATH. Therefore, right from the onset the realization of the MIC-CORE using ModGen was a reasonable option. However, keeping our principles concerning the design of the software in mind we made a decision against ModGen. The main reasons have been:

- The ModGen development platform requires Visual Studio 2005 which is a commercial and complex software.
- A WINDOWS system is mandatory for working with ModGen.
- The MicMac simulation model would have been predetermined to become a variant of LIFEPATH, with all its pre-set assumptions and variables.
- At the onset of the MicMac project the ModGen environment was not sufficiently documented from our point of view.

Relying on the results of our microsimulation survey we have drawn the conclusion that none of the existing tools (that we know) meets the demands we place on the MIC-CORE software.

3.3 "General" M&S Tools

There is a huge number of existing modeling and simulation tools around. Most of these are either bound to a certain computing infrastructure, modeling formalism, or even to a certain use case. Here those are of interest which are based on the discrete event modeling paradigm, because the multi-state model with its continuous time base is a discrete event model. Generally all tools could be used which support the simulation of discrete event based models, here we focus on LPs, DEVS-based, and agent models. μsik (Perumalla 2005) is a general discrete event simulation tool which has already been used for population forecasting (Satyabudhi and Onggo 2008). It has been created for supporting parallel and distributed simulations with "logical process" models - thus it contains more than we need on the modeling and execution level, as the logical process metaphor is more general than our application requires and from its metaphor it appears not to be entirely fitting, and a parallel distributed simulation is not what we were aiming at. In addition, it has been implemented in C which might cause some troubles if we are not aware of the operating systems on the machines of our users. There are several DEVS based tools around (e.g., DEVSJava, JDEVS, CD++, DEVS variants in JAMES II), supporting variants of the original DEVS (Zeigler, Praehofer, and Kim 2000) formalism. Not all of these support dynamic structures, as required. However, in addition the DEVS metaphor of reactive systems introduces modeling and simulation overhead, both of which we would like to avoid. Population dynamics is essentially about individuals and their population, and thus multi-agent M&S tools seem to fit here as well. However, multi agent M&S tools have a huge overhead as well: they typically incorporate at least agent mobility and communication issues, if not means for deliberation (Uhrmacher and Weyns 2009). If there is no tool to reuse we have to create an own one, something which is done very (if not far too) often. To decrease the amount of work to be done such a novel tool one should rely on existing, and hopefully well tested implementations and re-use the offered functionality. Such functionality is provided by some specialized M&S libraries as SSJ (L'Ecuyer, Meliani, and Vaucher 2002): they can be used to create new M&S tools by reusing commonly required algorithms and data structures. But using a library means that you still have to create a complete application from scratch. An alternative to libraries are frameworks. Frameworks are "abstract" applications for a certain domain which can be used to (easily) build specialized applications on top of these. They do not only provide the functionality of a library, they provide in addition a reusable combination of these, or even pre-defined workflows which only need to be activated.

4 IMPLEMENTATION IN JAMES II

The M&S framework JAMES II (available at `www.jamesii.org`, accessed April 2008) has been created at the University of Rostock from 2003 on. It is an open source project, and thus it is available at no costs, it runs on top of the Java platform (and thus on all machines for which a JVM exists), it is not bound to any modeling or simulation execution paradigm, nor to any formalism or language. As a framework it can be "transformed" into specialized applications meeting the needs of certain use cases. Any product created on top of the framework can be easily extended later on, e.g., by adding a new simulation algorithm to make use of multi-core CPUs – without any need to modify the application created. Consequently it can be used to create an application which fulfils all of the requirements listed above.

4.1 JAMES II: General Overview

JAMES II is a M&S framework based on the "Plug'n simulate" concept (Himmelspach and Uhrmacher 2007). As a framework for M&S it shall ease the creation of specialized M&S applications. Thereby these applications can either be created by extending the framework directly or by building the application on top of the framework (and thus by using the framework as a "M&S service"). The first option reduces the amount of work, and makes reuse rather easy, the second approach can imply an overhead, but means that you can create your application without any constraint. Technically the latter mostly means that you create your own GUI, whereby the first one means that you embed your application into the GUI of JAMES II. The framework basically consists of a lean core which provides the plug-in management systems, predefined, commonly used extension points, general basic functionality, support for different types of experiments (e.g., optimization, validation, ...), means to support distributed simulations, and an extensible GUI. The core has no integrated support for any modeling formalism, and there are no simulation algorithms in there as well. For each type of plug-ins to be installed (e.g., a modeling formalism, simulation algorithm, random number generator, partitioning algorithm) a special extension point needs to be provided, but then any number of plug-ins can be installed therefore. There are, e.g., event queue implementations in the core, which make it easier to create a plug-in providing a discrete event simulation algorithm, and, due to the plug-in concept the number and types of queues can be increased by everyone. JAMES II already provides a number of alternatives for most commonly used extension points, for random number generators, random distributions,

data sinks, modeling formalisms, simulation algorithms, and many more. In fact if you're going to use JAMES II to create your specialized modeling and simulation application you have to at least reuse or create a modeling formalism / language plug-in, and reuse or create at least one simulation algorithm. In general everything available can be reused, but you never need to do so. The default GUI can be used as a framework for the creation of more specialized GUIs, but a general GUI can only provide general model editing capabilities, e.g., an editor with syntax highlighting, and a very general experiment editor, and so on. So it might be useful to add a more specialized model editor, or even to create your own GUI from scratch. This means that you can integrate JAMES II into any other application, and just reuse the not visible functionality. Besides using JAMES II as a library of methods you can reuse complete processes as they are defined in JAMES II. Among these the experimentation process (Himmelspach, Ewald, and Uhrmacher 2008), which computes parameter combinations to be simulated with a given model, which takes care of model and simulation instantiation and instrumentation, and of the execution of a simulation on the available infrastructure.

4.2 Incorporation of Mic-Core

The MIC-CORE consists of two plug-ins a modeling plug-in, and a simulator plug-in.

4.2.1 The modeling plug-in for JAMES II

A model based on the MIC-CORE plug-in is being composed of two entity types: the *population* which comprises a set of *individuals*. Both are represented in the plug-in as interface and as class (`(I)Population`, `(I)Individual`). This allows (principally) to use alternative implementations of the interfaces at later stages, e.g., if models cannot be completely hold in the memory of the computer used or if alternative implementations promises a speed-up. Models are not completely implemented in Java here (only the two basic classes are coded): models are described by creating rate matrices which can stem from any source (such a source might be the statistical package R (See for more details `http://cran.r-project.org`, accessed February 2009.), and which are then read by an instance of the model reading mechanism of JAMES II. Thus, the model to be experimented with can be exchanged without the need to modify any line of code.

4.2.2 A simulator for MIC-CORE as a plug-in for JAMES II

The simulator is based on the classical "hold" loop. In an event queue all scheduled events for the individuals are hold (at most one event per individual). In each step we have to dequeue the event with the minimal time stamp, compute the state transition according to the model's state transitions, and enqueue a new event for this individual. The template for simulation algorithms in JAMES II requires to implement this functionality in the `nextStep` method. Everything else, i.e., run control, looping, a.s.o. is automatically handled by corresponding classes in JAMES II.

```
public void nextStep() {
  Event event = eventQueue.dequeue();
  List newEvents = computeTransition (event);
  for each newEvent in newEvents do
    eventQueue.enqueue (newEvent);
  }
}
```

The simple algorithm is given above. Events denote state transitions happening to specific individuals. The `computeTransition` method determines the new events to be scheduled, its functionality is explained in Section 2.2. Please note that individuals might be added or be removed from the population, and thus, if individuals are added more than one new event might be returned, or none, if an individual is removed. Subsequently, after a completed simulation run, the simulation output is stored using an adequate data sink plug-in that is provided by JAMES II.

4.3 Workflow of the microsimulation using MIC-CORE

Before performing the actual microsimulation, the model has to be specified and parameters (i.e., transition rates) need to be estimated. Furthermore, the analysis of a simulation output should be feasible as well. Both the modeling and the analysis should be done in a sophisticated manner, and it should be possible to employ a widely used statistical tool that is known to the community of potential users. In addition, such a software tool has to meet the principles regarding the MicMac software (see section 3.1). In order to cope with all requirements we have chosen R for implementing data preparation functions and summary functions for the simulation output. R is a free and open-source software environment for statistical computing and

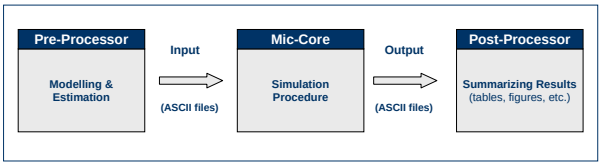

Figure 1: Workflow of MIC-CORE

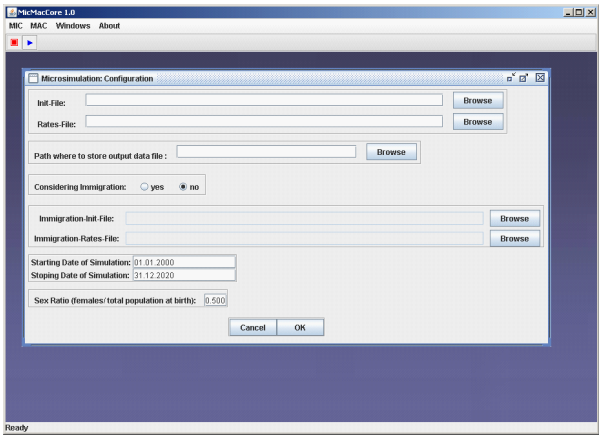

Figure 2: Using the GUI of the MicMac software the user can enter all instructions that the MIC-CORE needs to run a microsimulation

graphics that is equipped with up-to-date statistical methodology and high-quality plot options. Using R statistical models and corresponding estimation procedures that allow to derive the empirical input data for the simulation have been set up in the so called "Pre-Processor" of the MIC-CORE (van der Gaag, de Beer, and Willekens 2008). After data preparation the Pre-Processor builds up two ASCII based files in a well-defined format: one for the initial population and another one for the state space states and the related rates. We have implemented in R a comprehensive palette of instruments in order to evaluate and illustrate the output of a MIC-CORE run. We call this palette the "Post-Processor" of the MIC-CORE. Among others, the Post-Processor comprises the following features (Gampe and Zinn 2009): frequency tables of the states occupied at specific dates, population pyramids at specific dates, frequency distributions of the states occupied on January 1 of each year during the simulation period, analysis of first transition, analysis of origin and destination states, and identification of the most frequent ("typical") lifecourse. In order to meet our demands to provide a user friendly and easy to use tool both, the Pre-Processor and the Post-Processor, are self-contained sets of functions that can be used even without deeper knowledge of R. For the actual simulation run the MIC-CORE requests nothing more than the two input files prepared by the Pre-Processor and a time horizon for the simulation. By using a simple GUI which has been implemented for the MIC-CORE the user enters all required instructions to run a microsimulation (cp. Figure 2). Subsequently, after a completed simulation run, all individual attributes and life courses are stored. Everything related to the model execution is hidden from the user. Consequently, the user (we want to address) can easily employ the MIC-CORE. Detailed information on the usage of the MIC-CORE can be found in the corresponding manual (Zinn and Gampe 2009). In Figure 1 the workflow of the microsimulation using MIC-CORE is depicted. Here the data flows between Pre-Processor, MIC-CORE, and Post-Processor are displayed. Finally, it should be pointed out that the user is not forced to use Pre-Processor, MIC-CORE, and Post-Processor in combination. Alternative statistical software tools can be applied for preparing the simulation input data and for analyzing the generated output data as well.

5 AN EXAMPLE

5.1 "Production run" results

To illustrate the performance of the MIC-CORE we use the following example. We perform a simulation run without migration for a base population comprising 557,666 individuals. The individuals ages are in the interval from 0 to 98. The simulation will be executed over 20 years, starting on January 1, 2004 up to December 31, 2024. The considered state variables are (Variable values are written in parentheses after the variable names.): Sex (male, female), Marital status (never married, married, divorced, widowed), and Children ever born (0,1,2,4). A synthetic base population and transition rates that are based on simplified demographic information from different European data sources were used. Although the example uses imaginary data they are typical for contemporary Western European countries. The applied mortality rates are age- and sex-specific and vary over calendar time. The other transition rates are age- and sex-specific, but were held constant over calendar time. Figure 3 shows the transition rates for males to become married and the parity specific fertility rates for married females. We performed the example on an Intel(R) Core(TM) 2 Duo CPU with a 1.60GHz equipped with 2GB

memory 100 simulation runs. (We use a 'standard' machine as the user should be able to run a microsimulation on its own desktop computer.) The runs last between 119.18 and 121.03 seconds. (Approximatively 2 minutes per simulation run seems to be too short to advocate for a parallel execution of single runs here (in contrast to (Satyabudhi and Onggo 2008)). However, a coarse-grained execution (i.e., the execution of n simulation runs in parallel) can be of use, and this is automatically supported by JAMES II as well.) The Mersenne Twister random number generator was used for the generation of the random waiting times. Using the post-processor we are able to produce descriptive statistics of the output results. Some results are displayed in the Figures 4, 5, and 6.

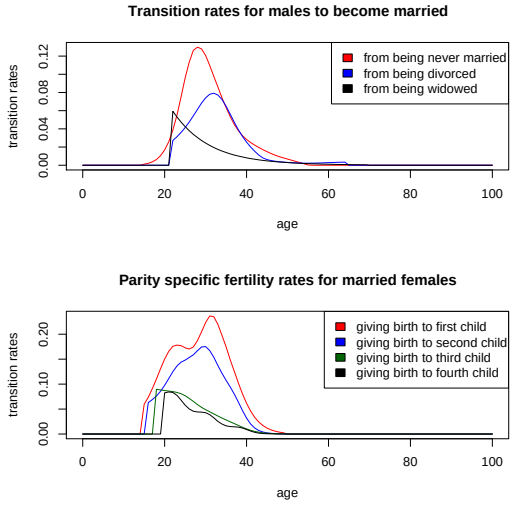

Figure 3: Some transition rates applied in the example

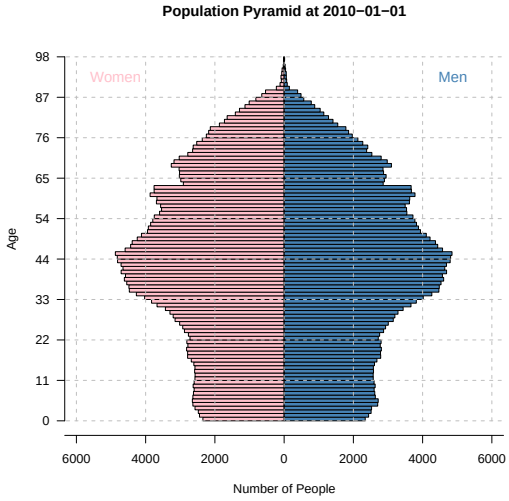

Figure 4: Population pyramid

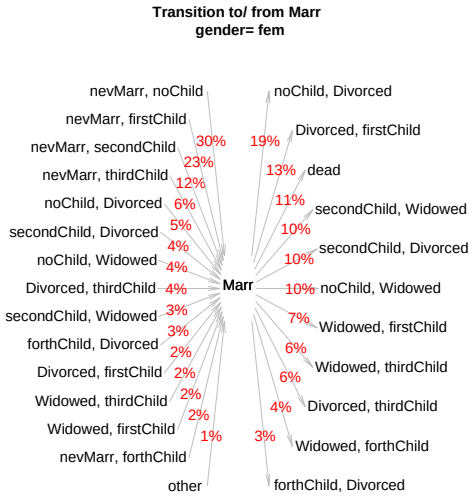

Figure 5: Relative frequency distributions of origin and destination states to/ from being married

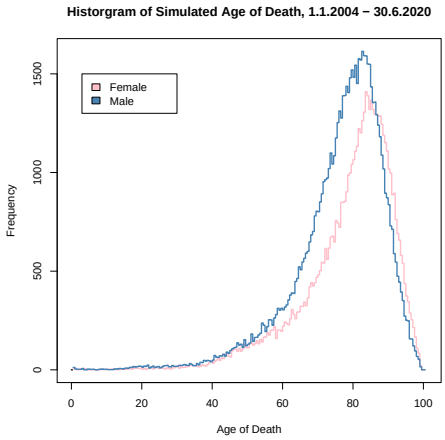

Figure 6: Distribution of deaths

5.2 Validation of the results

Validating the simulation output is good and useful practise. Besides basic validation of the simulation output, important hints for model improvement can be gained from careful analysis of the results, as several simplifying assumptions usually have to be made during the modeling process. Re-estimation of the empirical transition rates that we have used as input is the most basic validation step. Such a re-estimation can be performed applying occurrence-exposure rates (Keiding 1990). For smoothing these rates we employ an associated two-dimensional P-Splines methodology (Currie, Durban, and Eilers 2006) that has been implemented in an R draft package named `MortalitySmooth` (Camarda 2009). Concerning our example

the re-estimation of rates shows that the simulation causes consistent output. Some results are plotted in the Figures 7 and 8. Empirical rates along with re-estimated rates are presented. The first figure refers to female mortality over calendar time and age. The second figure is related to childless females who experience a transition from 'never married' to 'married'. As can be seen, the re-estimation confirms the validity of the results obtained from MIC-CORE.

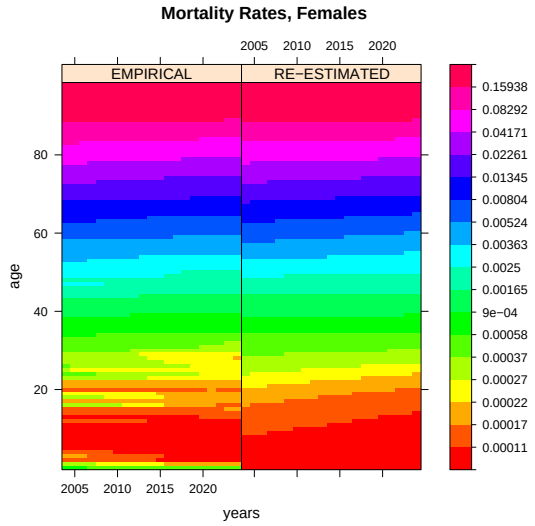

Figure 7: Re-estimation of female mortality rates

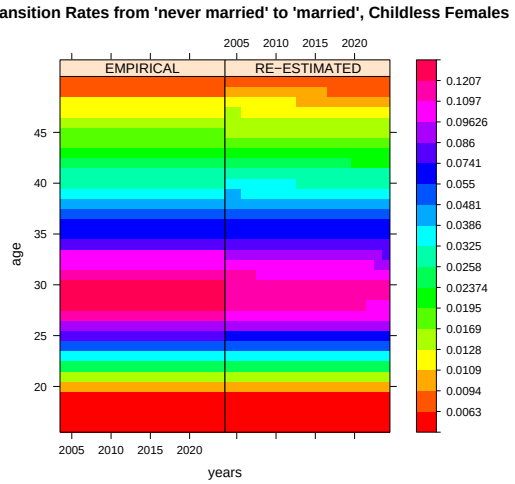

Figure 8: Re-estimation of transition rates of childless females who experience a transition from 'never married' to 'married'

6 CONCLUSION AND OUTLOOK

We have motivated the creation of a new tool, MIC-CORE, for forecasting population dynamics, and we have shown how to model for the tool. In addition we have conducted a small validation study to show that the tool seems to be valid, and we illustrated that the results can be computed in an acceptable time frame. The resulting tool meets the requirements. The new tool has been created based on JAMES II, and we had been interested in the question: How can (and did) JAMES II help on creating MIC-CORE? On the one hand JAMES II provides directly reusable functionality as *random number generators*, *random distributions*, *event queues*, and *data sinks*. This reduced the overall development complexity, especially due to the fact that for all these tests exist, and that they are already used for a variety of M&S scenarios, which increases the confidence in the implementations. In addition we reused the *simulation execution* process, including *model instantiation* and *model instrumentation*, as defined in JAMES II. This means that on developing MIC-CORE we had been able to focus on the model description, the simulation algorithm, and the new and standalone GUI of MIC-CORE. Although there had not been a sufficient documentation of JAMES II, nor an official release, MIC-CORE was created by the MPI group with only a little technical help. The JAMES II group learned a lot about a user's perspective on a framework, which will have an influence on the future documentation of the project. There is a variety of opportunities how to extend the model or the simulation part which will make MIC-CORE future safe. Some of these are very easy to have and depend on the above mentioned flexibility of JAMES II, e.g., to use a database instead of the currently used file based data export or to use coarse-grained simulation on multi-core/multiple CPUs. For each of these changes only a parameter has to be changed in JAMES II. Other extensions might imply some more work, e.g., to realize a new multi-threaded simulation algorithm. However, in this case only the simulation algorithm has to be exchanged (partitioning strategies can than be automatically re-used in addition): everything else of MIC-CORE remains constant, and one can switch between the algorithms later on easily. It is a well-known fact that the performance of algorithms differ depending on the problem at hand. JAMES II allows to have any number of alternative algorithms and data structures available and we are currently working on an automatic selection of efficient algorithms (Ewald, Himmelspach, and Uhrmacher 2008). This mechanism can be exploited as well, and thus MIC-CORE can adapt to model and hardware characteristics to fulfil the requirement of an efficient execution, e.g., on "normal" desktop workstations - and the users will even not get aware of this at all, and thus the system remains usable by non-specialists as well. The model description can and shall be extended with "linked lives" in the future. Which types of "linked lives" are of interest, and how these can be integrated in a well-defined manner, and how this relates to established formalisms, is currently under examination.

REFERENCES

Andersen, P., and N. Keiding. 2002. Multi-state models for event history analysis. *Statistical Methods in Medical Research* 11 (2): 91–115.

Bartholomew, D. J. 1973. *Stochastic models for social processes*. 2nd ed. John Wiley & Sons.

Brown, L., and A. Harding. 2002. Social modelling and public policy: Application of microsimulation modellingin Australia. *Journal of Artificial Societies and Social Simulation* 5 (4): <http://jasss.soc.surrey.ac.uk/5/4/6.html>.

Camarda, C. 2009. *MortalitySmooth: Smoothing Poisson counts with P-splines*. MPI Rostock. Draft R Package Version.

Currie, I., M. Durban, and P. Eilers. 2006. Generalized linear array models with applications to multidimensional smoothing. *Journal of the Royal Statistical Society B* 68 (Part 2): 259–280.

Duee, M. 2005. La modélisation des comportements démographiques dans le modèle de microsimulation DESTINIE. Technical report, INSEE, Institute National de la Statistique et des Etudes Economiques, Paris.

Ewald, R., J. Himmelspach, and A. M. Uhrmacher. 2008. An algorithm selection approach for simulation systems. In *Proceedings of the 22nd ACM/IEEE/SCS Workshop on Principles of Advancedand Distributed Simulation (PADS 2008)*, 91–98. Rome, Italy: IEEE Computer Society.

Favreault, M., and K. Smith. 2004. *A primer on the dynamic simulation income model (dynasim3)*. The Urban Institute, Washington, D.C.

Galler, H. 1997. Discrete-time and continuous-time approaches to dynamic microsimulation (reconsidered). Technical report, NATSEM - National Centre for Social and Economic Modelling, Faculty of Management, University of Canberry.

Gampe, J., and S. Zinn. 2007. Description of the microsimulation model. Technical report, MPIDR, Rostock.

Gampe, J., and S. Zinn. 2009. *Manual of the MicMac post-processor*. MPIDR, Rostock.

Gampe, J., S. Zinn, F. Willekens, and N. van den Gaag. 2007. Population forecasting via microsimulation: Software design of the MicMacProject. In *Work Sessions on Demographic Projections*: European Communities.

Gilbert, N., and K. Troitzsch. 2008. *Simulation for the social scientist*. 2nd ed. Open University Press.

Hammel, E., C. Mason, and C. Wachter. 1990. *SOCSIM II: A sociodemographic microsimulation program rev. 1.0. operatingmanual*. Regents of the University of California.

Harding, A. 2007. Challenges and opportunities of dynamic microsimulation modelling. Plenary paper presented to the 1st General Conference of the International Microsimulation Association, Vienna, 21 August 2007.

Himmelspach, J., R. Ewald, and A. M. Uhrmacher. 2008. A flexible and scalable experimentation layer for JAMES II. In *Proceedings of the Winter simulation conference*, ed. S. J. Mason, R. R. Hill, L. Mönch, and O. Rose, 827–835: Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Himmelspach, J., and A. M. Uhrmacher. 2007. Plug'n simulate. In *ANSS '07: Proceedings of the 40th Annual Simulation Symposium*, 137–143. Washington, DC, USA: IEEE Computer Society.

Holmer, M., A. Janney, and B. Cohen. 2009. *PENSIM overview*. Policy Simulation Group and the U.S. Department of Labour.

Hougaard, P. 1999. Multi-state models: A review. *Lifetime Data Analysis* 5 (3): 239–264.

Imhoff, E., and W. Post. 1998. Microsimulation methods for population projection. *Population: An English Selection, New Methodological Approaches in SocialSciences* 10 (1): 97–138.

Kalbfleisch, J., and R. Prentice. 1980. *The statistical analysis of failure time data*. New York: Wiley.

Keiding, N. 1990. Statistical inference in Lexis diagram. *Philosophical Transactions of the Royal Society of London: Physical Sciences and Engineering* 332 (1627): 487–509.

King, A., H. Baekgaard, and M. Robinson. 1999. Dynamod-2: An overview. Technical report, NATSEM, National Center for Social and Economic Modelling, University of Canberra.

Klein, J., and M. Moeschberger. 2003. *Survival analysis. Techniques for censored and truncated data*. 2nd ed. New York: Springer.

Klevmarken, N., and P. Olovsson. 1996. Direct behavioral effects of income tax changes - simulation with the Swedish model MICROHUS. In *Microsimulation and Public Policy*. Amsterdam: North-Holland-Elevier.

L'Ecuyer, P., L. Meliani, and J. Vaucher. 2002. SSJ: a framework for stochastic simulation in Java. In *Proc. of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 234–242: Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Mayer, K. U., and N. B. Tuma. (Eds.) 1990. *Event history analysis in life course research*. University of Wisconsin Press.

Morrison, R., and B. Dussault. 2000. Overview over DYNACAN: A fully-fledged Canadian actuarial stochastic model designed for the fiscal and policy analysis of social security schemes. http://www.actuaries.org/CTTEES_SOCSEC/Documents/dynacan.pdf. [accessed February 2009].

NIDI 2006, June. MicMac newsletter. http://www.nidi.knaw.nl/en/micmac/newsletter/newsletter_1/micmac-newsletter-01.pdf.

O'Donoghue 1999. Dynamic microsimulation: A methodological survey. *Brazilian Electronic Journal* 4 (2).

Orcutt, G. 1957. A new type of socio-economic system. *Review of Economics and Statistics* 39 (2): 116–123.

Orcutt, G., M. Greenberger, J. Korbel, and A. Rivlin. 1961. *Microanalysis of socioeconomic systems: A simulation study*. New York: Harper & Row.

Perumalla, K. 2005. μsik: A micro-kernel for parallel/distributed simulation systems. In *ACM/IEEE/SCS Workshop on Parallel and Distributed Simulation (PADS)*. Monterey, CA: IEEE Computer Society Press.

Rubinstein, R., and D. Kroese. 2008. *Simulation and the Monte Carlo method*. 2nd ed. Series in Probability and Statistics. Wiley.

Satyabudhi, B., and S. Onggo. 2008. Parallel discrete-event simulation of population dynamics. In *Proceedings of the 2008 Winter Simulation Conference*, ed. S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, and J. W. Fowler, 1047–1054: Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Sauerbier, T. 2002. UMDBS - a tool for dynamic microsimulation. *Journal of Artificial Societies and Social Simulation* 5 (2): <http://jasss.soc.surrey.ac.uk/5/2/5.html>.

Spielauer, M. 2006. The "lifecourse" model, a competing risk cohort microsimulation model: source code and basic concepts of the generic microsimulation programming language ModGen. http://www.demogr.mpg.de/papers/working/wp-2006-046.pdf.

Statistics Canada 2004. The lifepath microsimulation model: An overview. http://www.statcan.gc.ca/spsd/LifePaths.htm. [accessed February 2009].

Statistics Canada 2009. Modgen version 9.0.10: Developer's guide. http://www.statcan.gc.ca/spsd/ModgenDev_EN.pdf. [accessed February 2009].

Uhrmacher, A. M., and D. Weyns. (Eds.) 2009. *Multi-agent systems: Simulation and applications*. Taylor and Francis.

van der Gaag, N., J. de Beer, and F. Willekens. 2005. Combining micro and macro approaches in demographic forecasting. In *Work Sessions on Demographic Projections*: European Communities.

van der Gaag, N., J. de Beer, and F. Willekens. 2008. The MicMac pre-processor. Technical report, NIDI, The Hague.

van Imhoff, E. 1990. The exponential multidimensional demographic projection model. *Mathematical Population studies* 2 (3): 171–181.

Willekens, F. 2005, May. Bridging the micro-macro gap in population forecasting. MicMac project proposal, NIDI, The Hague.

Willekens, F. 2007. Continuous-time microsimulation in longitudinal analysis. Technical report, NIDI, The Hague.

Willekens, F., J. de Beer, and N. van der Gaag. 2007a. Report on input data requirements of MIC. Technical report, NIDI, The Hague.

Willekens, F., J. de Beer, and N. van der Gaag. 2007b. Report on input data requirements of MIC. Technical report, NIDI, The Hague.

Wolf, D. 1986. Simulation methods for analyzing continuous-time event-history models. *Sociological Methoddology* 16:283–308.

Wolf, D. 2001. The role of microsimulation in longitudinal data analysis. *Special Issue on Longitudinal Methodology, Canadian Studies in Population* 28 (2): 313–339.

Zeigler, B. P., H. Praehofer, and T. Kim. 2000. *Theory of modeling and simulation*. 2nd ed. London: Academic Press.

Zinn, S., and J. Gampe. 2009. *MicCore user's guide*. MPIDR, Rostock.

AUTHOR BIOGRAPHIES

SABINE ZINN is a Ph.D candidate at the MPIDR and the University of Rostock. She received a diploma in business mathematics from the University in Jena. She is interested in microsimulation and statistical programming. Her email address is <zinn@demogr.mpg.de>.

JAN HIMMELSPACH is a post doc in the Computer Science Department at the University of Rostock. He received his Ph.D. in Computer Science from the University of Rostock. His research interest is on software engineering for modeling and simulation, credibility of modeling and simulation, and on efficient modeling and simulation solutions. His email address is <jh194@informatik.uni-rostock.de>.

JUTTA GAMPE is the Head of the Laboratory of Statistical Demography of the MPIDR. She received her Ph.D in Statistics from the TU Berlin. Her research interest is in statistical demography and related topics. Her email address is

<gampe@demogr.mpg.de>.

ADELINDE M. UHRMACHER is an Associate Professor at the Department of Computer Science at the University of Rostock and head of the Modeling and Simulation Group. Her research interests are in modeling and simulation methodologies and their applications. Her email address is <lin@informatik.uni-rostock.de>.