TITLE PAGE

Approaching parallel computing to simulating population dynamics in demography

Cristina Montañola-Sales[1,4], Bhakti S. S. Onggo[2], Josep Casanovas-Garcia[1,4], Jose María Cela-Espín[3,4], Adriana Kaplan-Marcusán[5]

1) Departament d'Estadística i Investigació Operativa, Universitat Politècnica de Catalunya - BarcelonaTech, Barcelona, Spain, pone: +34 93405 4190, email: cristina.montanola@upc.edu *(corresponding author)*

2) Department of Management Science, Lancaster University Management School, Lancaster, United Kindom

3) Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya-BarcelonaTech, Barcelona, Spain

4) Barcelona Supercomputing Center, Barcelona, Spain

5) Departament d'Antropologia Social i Cultural, Universitat Autònoma de Barcelona, Spain

Approaching parallel computing to simulating population dynamics in demography

Abstract

Agent-based modelling and simulation is a promising methodology that can be applied in the study of population dynamics. The main advantage of this technique is that it allows representing the particularities of the individuals that are modeled along with the interactions that take place among them and their environment. Hence, classical numerical simulation approaches are less adequate for reproducing complex dynamics. Nowadays, there is a rise of interest on using distributed computing to perform large-scale simulation of social systems. However, the inherent complexity of this type of applications is challenging and requires the study of possible solutions from the parallel computing perspective (e.g., how to deal with fine grain or irregular workload). In this paper, we discuss the particularities of simulating populating dynamics by using parallel discrete event simulation methodologies. To illustrate our approach, we present a possible solution to make transparent the use of parallel simulation for modeling demographic systems: Yades tool. In Yades, modelers can easily define models that describe different demographic processes with a web user interface and transparently run them on any computer architecture environment thanks to its demographic simulation library and code generator. Therefore, transparency is provided by by two means: the provision of a web user interface where modelers and policy makers can specify their agent-based models with the tools they are familiar with, and the automatic generation of the simulation code that can be executed in any platform (cluster or supercomputer). A study is conducted to evaluate the performance of our solution in a High Performance Computing environment. The main benefit of this outline is that our findings can be generalized to problems with similar characteristics to our demographic simulation model.

Keywords: agent-based simulation, simulation tool, demography, population dynamics, high performance computing, transparency

1. Introduction

Changes in our society driven by social, economic, environmental and technological developments have created a challenge for policymakers. As a response to this, the European Union's Horizon 2020 program has established a research agenda centred on societal challenges like ageing, energy saving, smart transport, secure internet, inclusion and preservation of cultural heritage. Many of these challenges can be overcome by using analytic tools that rely on right projections of future population. Simulation is one of the tools with a lot of potential in dealing with population dynamics and population projection. Particularly, discrete event simulation has long been used as a simulation methodology for capturing the behaviour of a system using a sequence of discrete events during time. Events occur at a particular time instant, producing a change in the simulation state of the system (Averill and Kelton, 2007). This methodology is particularly useful for reproducing the activities and interactions of individuals, social patterns, and population movements on a local and global scale. However, in a large context these simulations can be difficult to obtain due to the number of agents and interactions involved and the complexity of those interactions among them and their environment. Therefore, parallel simulation techniques can further provide support to manage the execution of complex social simulation models.

Traditionally, parallel simulation has been applied in military and network simulations. However, since 2005 there has been an increase in the number of papers reporting on parallel simulation applications outside traditional areas. The reason behind is that the increase of power consumption and clock speed improvements following the Moore's law began to stall in 2004. As a result, hardware manufacturers switched to the multicore technology, situation that possibilitates the acess to large amounts of computing power and memory. Therefore, in the petascale era performance increases come through parallelism. This change of scenario was key to raise the use of parallel computing, since applications continuously require to process models with a growing number of parameters and larger datasets, along with a competitive simulation time. In this context, the programmer not only has to define the simulation model and algorithms to use, but also how to distribute data and work among the parallel processing elements in the architecture.

Generally in parallel discrete event simulation, the whole simulation state is partitioned in multiple subportions of the sate (logical processes) which execute independently of each other. Each logical process has a private clock and executes without any shared portion of memory. Logical processes communicate through event exchange and are autonomous in the sense that they can determine for themselves how to process their received events. Since the beginning of petascale systems, we have seen applications of parallel discrete event simulation outside traditional areas such as plasma physics (Tang et al., 2005), in the realm of biological science (Lobb et al., 2005), manufacturing (Lan and Pidd, 2005), traffic simulation (Yoginath and Perumalla, 2008), electromagnetics (Bauer-Jr. et al., 2009), or archaelogy (Rubio-Campillo et al., 2012). The current number of applications is encouraging, although it is still far from ideal.

Therefore, there is a need of more works to promote the use of parallel computing techniques in real applications, particularly in the social sciences. The analysis of more detailed models of social behavior would help us understand, for example, how pandemics spread allowing health policy-makers and planners better estimate the effectiveness of different strategies to limit the spread of diseases. High-performance parallel computing is particulary important in the context of such what-if analysis where the production of simulation outputs on time is critical for decision making. Moreover, social scientists and policy-makers are not necessarily close to computer science methodologies to speed up their simulations. Therefore, this is a perfect context to develop methodologies and frameworks to make available the use of parallel simulation transparently. In this context, we refer to transparency taking the definition of Solcany and Safarik (2002) and Pellegrini (2015) in the sense that (i) no modification of the original (sequential) simulation model is needed, and (ii) users do not have to worry about the detail on how to harness the parallel computer power. Since the development of parallel programs is estimated to cost ten times as much as developing sequential programs (Rajaei and Ayani, 1993), transparency is important. Therefore, with a framework that provides transparent parallel solutions the development speed will increase and a reduced execution time of simulation models will be obtained (Rajaei, 1992). Moreover, modelers can concentrate more on the actual model definition, and reduce the effort on model development and parallel intricate mechanisms (Rönngren et al., 1996).

In the application area of simulating complex social systems, Agent-Based Modeling (ABM) is a bottom-up modeling approach that has gained popularity in recent years. It allows gaining insights of social complexity. An agent-based model allows the simulation of the dynamics of a population by controlling the characteristics and behavior of each individual of the system (Ferrer et al., 2009). Moreover, agent-based modeling is particularly useful for projecting a population by answering "what if" questions such as the effect of a certain policy on the spread of a disease in a target group. It possibilitates modelling the impact of personal decision making processes in strategic planning or government policies. Although the population projection is a simplification and uncertain representation of the modelled people, it is often used as an input to models utilized for planning and policy making. Demography is an area of study which has greatly contributed to the study of population projection to guide policy analysis on societal planning (Rees, 2009).

In the past few years, supported by the advances in computer technology and the availability of data at micro level (individuals), the use of micro-level simulation models in population projection has become more widespread. The main advantage of this approach is that individual-specific explanatory variables can be included in the model so the main advantage is that it opens up new research fields. For example, we may include factors such as age, education level, salary group and ethnicity to model the number of children that an individual female will have. This capability has attracted quantitative social science researchers and practitioners such as anthropologists, historians, and demographers to investigate the potential use of micro-level simulation models in their research. In addition, the projected population is often used as an input to policy models which at the same time are often taken as the basis for government policies in areas such as labour market, education, healthcare, social welfare and taxation.

In the context of demographic studies, agent-based models have been used for performing simulations. Some of the relevant works have been published in a book titled “Agent-Based Computational Demography” (Billari & Prskawetz, 2003). In this paper, we propose a solution to bridge the gap between the need for efficient parallelism exploitation, and the complexity of developing applications of population dynamics for demography. Our proposal uses an agent-based approach to conduct large-scale demographic simulations. To support its usage, we provide a web-based user interface which will also offer help to modelers who may not have any parallel programming skills. Currently, there are several tools that support the development and execution of generic agent-based models (see for example Collier, 2001; Mason; Minar et al., 1996; Tisue & Wilensky, 2004). However, desktop agent-based modeling tools may not scale well for large-scale simulations. Recently, some efforts have been made in this direction by using High Performance Computers to distribute the workload of agent-based simulation between a number of processors (see Collier and North, 2012; Cordasco et al., 2011; Rubio-Campillo, 2014). The work presented in this paper contributes to the current body of knowledge addressing specifically large demographic simulations.

The rest of this paper is organized as follows. In Section 2 an overview of related work in demographic simulation is presented, including some previous work on agent-based simulation in demographics. Section 3 discusses the challenges of large-scale agent-based models for population dynamics in parallel environments. Our framework for parallel demographic simulation then follows in Section 4. Section 5 shows an evaluation of the performance results of the simulation tool. Finally, the concluding remarks and suggestions for further work are discussed in Section 6.

2. Related works

An increase in the popularity of simulation for population projection has arised during the past decade. One of the commonly used paradigms in demographic simulation is micro-simulation, which has been widely applied in the field of migratory movements or human reproduction (Billari et al., 2003). The initial work in micro-simulation was first introduced by Orcutt (1957). In this paradigm, modelers have to specify a random sampling process for each individual at each simulation time point, to determine its state at the next time point. At one end, the sampling process requires a simple random sampling. At the other end, it may require a sophisticated regression model. Despite the complexity of these processes, most micro-simulation tools have been built for certain public policies. Examples include LABORsim for policies related to labour supply in Italy (Leombruni and Richiardi, 2006) and Pensim2 for the British pension system (O'Donoghue and Redway, 2009). SOCSIM (Hammel and Wachter, 1990) is one among the few generic micro-simulation tools for demography.

Dahlen (2009) and Zinn et al. (2009) developed generic open-source micro-simulation alternatives. These tools have proven to be useful in evaluating decission making processes in public policy analysis. However, they require a model detail that is difficult to achieve (randomness, countless parameters, data reliability and quality).

System dynamics is another commonly used modeling paradigm in developing demographic simulation models. Unlike micro-simulation, system dynamics does not keep track changes in each invidual' state but focuses more on the group of individuals and the rates of them shifting from one state to another. System dynamics is generally used to examine the complex feedback systems and the mutual interactions in the system over time. Important works in this area include the World Dynamics (Forrester, 1971) and World3 population model (Meadows and William, 1972; Meadows and Randers, 2004). A demographic model based on system dynamics is often used as a component in policy modelling. For instance, Ahmad and Billimek (2005) developed a system dynamics model which analyzes policies to reduce the harmful effects of tobacco on population health. Saysel et al. (2002) developed a system dynamics model to evaluate policies on various environmental issues such as water distribution management and agricultural pollution.

Similar to microsimulation, discrete-event simulation keeps tracking the individuals from their arrival in the system (through births and migrations) to their departure (through deaths and migrations). However, discrete-event simulation does not inspect each individual at each simulation time point. Instead, it evaluates an individual only when the state of the individual changes, thus increasing simulation performance. Most discrete-event demographic simulation models are applied to fields such as healthcare and epidemiology. For example, Rauner et al. (2005) proposed a discrete-event simulation model to study the effectiveness of intervention programs to reduce the vertical HIV transmission. A number of attempts have targeted to build large-scale epidemiological simulation models. The main purpose is to understand the spread of global epidemics which may include analysis of a large number of individuals. For instance, Montañola-Sales et al. (2015); Prats et al. (2016) showed how discrete-event models can be used to evaluate public policies in the transmission of tuberculosis in big cities. Eubank (2002) and Rao and Chernyakhovsky (2008) showed with their development of specialized simulation tools that parallel discrete-event simulation was required for large-scale epidemiological models.

Demographic simulation has also been tackled by means of agent-based models. Read (1998) used an agent-based model to explore the interrelation between the demographic system and the cultural system in an artificial society of hunter-gatherers. Heiland (2003) explained migration flows from East to West Germany from 1989 to 1991 by using agent-based simulation. Also Benenson et al. (2003) took the same approach to understand residential dynamics in Yaffo (Israel). Among recent works in this area, agent-based extensions of a spatial microsimulation model of demographic change have been proposed by Wu & Birkin (2012) for projecting the student migration and mortality in Leeds (UK). Geard et al. (2013) showed an example of how agent-based modelling can be applied to create a synthetic population able to describe basic demographic processes and explore their interaction with patterns of infection and immunity. Kniveton et al. (2011) proposed an agent-based approach to understanding environmental migration in Burkina Faso with the purpose of assisting policy makers. Silverman et al. (2013) presented an example which uses agent-based models to evaluate family structures changing in the UK population and the health care provision.

As in previous works, our proposed simulation tool implements a set of agent-based demographic models to explain population dynamics. However, we use parallel environment in order to take advantage of High Performance Computing capabilities to run large-scale simulations. Although in agent-based simulation some attempts have already been made to explore demographics, none of them can cope with large scenarios. Despite the existence of other tools that deal with large-scale agent-based models, their interface is not specifically designed to model demographic human behavior neither they target demographers and policy makers.

3. Challenges of parallel social simulation

To understand the complex nature of social systems, social researchers have a range of methodologies available. Simulation is among them. The intrinsic dynamic nature of real-world social phenomena may easily lead to simulations too slow to provide the needed insights for researchers (Allen, 2011). That is where parallel computing enables to manage realistic models. However, the distribution of social models among computers in a network of nodes is not an easy task. Social models are irregular applications with particular characteristics such as fine-grained communication or no straightforward solutions to balance the workload. Scalability needs to be addressed, although there is no consensus on dealing with the difficulties it encounters on agent-based models (Hybinette et al., 2006; Rubio-Campillo, 2014; Tesfatsion, 2002).

There are multiple situations where parallel simulation can leverage the execution time of agent-based simulations. On first instance, it is especially valuable in cases where the execution time is too slow as a result of including a large number of agents and having a simulation time very small. The duration of a large simulation will depend not only on the processor speed but also on the capacity. If the execution requires a bigger memory space than the memory capacity of a single processor, it will cause numerous operations on memory swapping which repercute on an increased simulation time. Although most of the agent-based models in the literature are barely large-scale, advances in social fields could make the simulation of these scenarios more necessary. On second instance, including enriched decision-making processes in the agent model, such as rational behaviour or cognitive and psychological processes, can require higher computing demands. On third instance, parallel simulation offers a solution when exploring emergent properties that a small-scale variant of the model is not able to cope with. For instance, parallel simulation might be the only solution in the case the emergent property is linked to the number of interactions at a given time step (Mithen and Reed, 2002; Rubio-Campillo et al., 2012). On fourth instance, it is important to take into account that we are dealing with non-linear, dynamic systems with high uncertainty and notable degree of stochasticity. As a consequence, the exploration of the model's parameter space would be required to obtain calibrated results. This parameter sweep might easily imply a considerable number of runs. Parallel simulation minimizes the time needed to perform this task.

There are several aspects which might particulary affect the scalability of agent-based models: the complexity of agents, the topology of communications, and the representation of the environment. Advanced approaches to human modelling might represent a high demand in computer power and be crucial to the parallelization of social simulation models. Approaches in Artificial Intelligence such as the Belief, Desires, Intentions scheme (BDI) can be used for enriching the individual raisoning (Bratman, 1999), with the consequent load of computation power.

Scalability can be highly vulnerable to the complexity and topology of communications between agents. The number of communications in the system depends mainly on the distribution of space, the implementation of agents, and the number of their movements across the environment. A common approach is to perform a spatial partitioning, which is strongly dependent of the topology (either static or dynamic) of agents. However, there exist models were space is not important, such as in the case of the study of social mechanisms like evolution of paternal care (Salgado, 2013). Therefore, it is important to understand the logic of the model to efficiently divide the environment across computer processors. For instance, in the case of migrations flows an efficient approach could be adapting the space partitioning to these population movements and considering the balance between regions and the 'bottlenecks' migrations (communications) they may produce.

Another issue that might affect scalability is the representation of the environment, which can be as simple as dividing the model in different parts (regions) or as complex as a Graphical Interface System (GIS). A GIS can not only contain data at a geographical scale but also alternative data such as culture, political ideology or religion (Castle and Crooks, 2006). In agent-based systems, each agent needs to gather knowledge from the environment, as well as from other agents, in order to execute its decision making processes. Agents may also modify the environment. The variety of environments might lead to different design solutions, from the distribution of space across processors in the simplest scenario (see for example, Parry and Bithell (2012)) to more sophisticated variations. Although there have been some initiatives to automatize the parallelisation of agent-based simulations (Coakley et al., 2012; Kurowski et al., 2009), overall, the nature of the problem and the properties of the computer platform will often guide the method to split the simulation execution.

Therefore, scalability of social agent-based models is not a trivial matter and requires an interdisciplinary effort. On one hand, computer scientists need to study the particularities of the social domain to successfully participate in the model design. On the other hand, social researchers need to be aware of the computational challenges the model generates at different layers (availability of computational resources, experiment design, model scale, and so on).

4. A solution for parallel demographic simulation: Yades framework

This paper presents an example on bringing transparency to parallel discrete-event simulation by the design and implementation of Yades (*Yet Another Demographic Simulator*), a parallel demographic agent-based simulation tool. Yades uses an agent-based approach to model fertility and birth, mortality, economic status, marital status, and migration demographic processes which permits individuals to flexibly move and interact in a geographical environment:

1. It uses a rich set of attributes which originate not only from census or surveys data sources, but also from behavioural rules that help to overcome some data-related limitations of over-reliance on purely statistical information. Individuals' attributes change over time due to their demographic evolution.
2. There is no central unit that controls interactions or behaviours of the population
3. Agents behave autonomously according to their own rules
4. Agents do not have global information, they take actions according to simple rules that are based on local knowledge

Yades has three components: a web user interface, a demographic simulation library and the simulation code generator. The web user interface allows demographic modelers to specify demographic model components in a number of representations familiar to demographers such as regression and statistical distribution function. The simulation code generator can produce the corresponding C++ code that is linked to the demographic simulation library which uses a scalable parallel discrete-event simulation engine. The generated code is ready for compilation using a target C++ compiler. The demographic simulation library supports both sequential and parallel execution of the simulation model. The framework is shown in Figure 1**¡Error! No se encuentra el origen de la referencia.**.

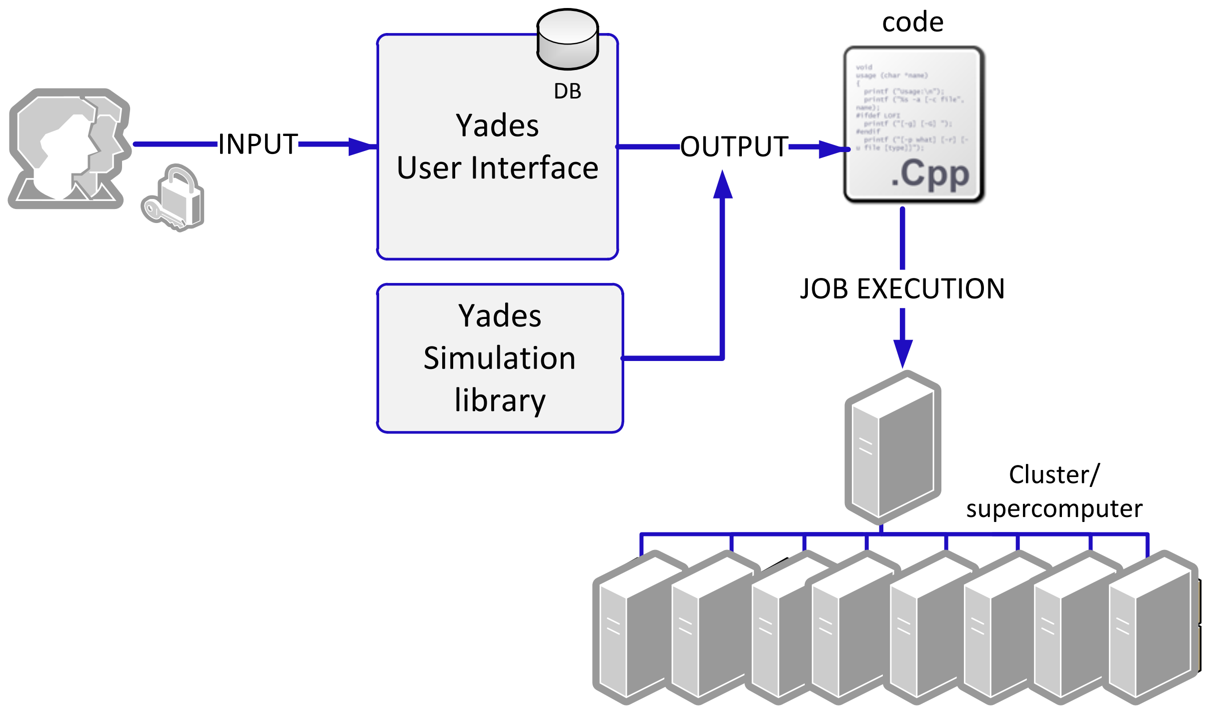


Figure 1. Yades framework for paralell demographic simulation

Yades models the life course of individuals in an environment formed by a set of regions through five demographic components which are typically used on demography (Hinde, 1998): birth (fertility), change in economic status and marital status, migration, and death (mortality). The fertility component determines whether a female individual will give birth, based on the characteristic of the female individual and the current calendar time. Yades includes the option to use age to determine the probability of having a child (age-specific fertility), to take into account the number of children a female has already had (parity-specific fertility), to focus on the time between each birth (birth spacing), and their combinations. Similarly, modelers can use the characteristic of an individual and the current calendar time to determine a new economic status of that individual (generally with a state diagram). Complex models will include explanatory variables such as an individual's characteristics, the characteristics of the individual's family and external socio-economic factors. A new marital status can be modelled based on the characteristics of the individual (or individuals for a couple) and the current calendar time. If the new status is either married or cohabitating, modelers need to define the criteria that will be used to match the individual to another individual from the list of prospective partners (match making function). Similar to economic status, most marital status models use state-transition diagrams to represent the possible changes in status. Modelers also need to specify a model that is used to determine whether a family unit is going to migrate (either domestically or internationally). A simple model uses a simple random sampling to decide whether an individual is going to migrate and to determine the new area. A more complex model employs a combination of individual-specific factors, family-specific factors, region-specific factors and other external factors to explain the individual's decision to migrate to a new place. Finally, in the mortality component, modelers need to model the time when an individual will die based on the characteristics of the individual. Commonly used methods, such as life table and survival function can be used for the mortality component.

There have been different works in the literature about providing transparency to parallel discrete-event simulation. For example Rönngren et al. (1996) presented a method to implement a transparent incremental state saving mechanism in an optimistically synchronized parallel discrete event simulation system based on the Time Warp (optimistic) mechanism that proved to be useful in the simulation of cellular phone systems. Solcany and Safarik (2002) showed a simulator design for build parallel simulation models with lookahead transparently without any substantial penalty performance due to the transparency. Pellegrini et al. (2012) extended of the traditional Time Warp synchronization protocol for parallel/distributed simulation by handling global variables in a more transparent way, saving event communications across logical processes. In our approach, we propose to increase transparency to social simulations by providing an environment where users can benefit from the facilities of parallel computing and obtain faster execution time without the need to be concerned about specific parallel mechanisms.

There are several run-time support libraries for running parallel discrete-event simulations such as Root-Sim (Alessandro Pellegrini & Quaglia, 2014), ROSS (Carothers, Bauer, & Pearce, 2002), Warped (Martin, McBrayer, & Wilsey, 1996) or μsik (Perumalla, 2005). As a case study, we chose μsik simulation library for parallel discrete-event simulation (Onggo, 2008, 2010). It has been used in multiple projects and has shown a good performance in the classical benchmark Phold (Perumalla, 2005). In μsik, models adopt the process interaction world-view; hence a simulation model is formed by a set of interacting (logical) processes (LP) hosted on each processor. Logical processes are fully autonomous entities that can determine for themselves how to process their received events. Therefore, we used µsik (logical) processes to implement agents in our demographic agent-based model.

Logical processes communicate through events with the standardized protocol Message Passing Interface (MPI) (Pacheco, 1997), also used for synchronization. Multiple logical processes are mapped onto a physical process (µsik kernel) that is run on top of a processing element. A machine can have more than one processing element (e.g., in multi-core architecture). Figure 2 shows the layers of software needed to implement the parallel simulation tool using µsik. µsik kernel provides services to locate all µsik processes, efficiently communicate events with other remote processes through multicast exchanges, and collectively achieve the correct time-ordered processing of events. This scheme is common in the parallel discrete-event simulation literature. A drawback of this approach is that the communication costs become a wasted overhead if the event is later retracted, due to a user request or a process of event cancellation.

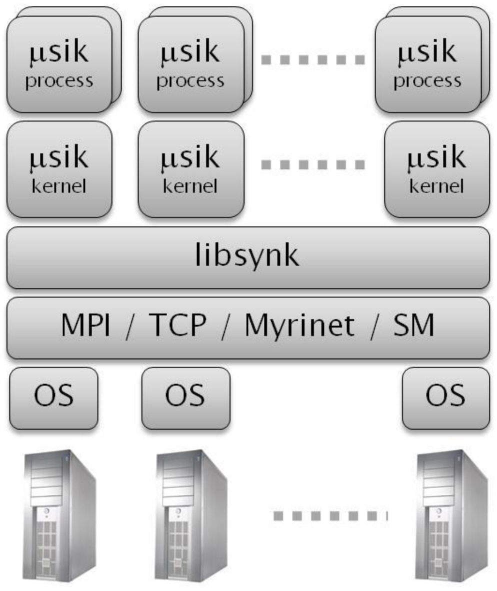


Figure 2. Parallel simulation using µsik

In µsik, event lifecycle consists of a simulation process that allocates and schedules an event. Then, the receiver starts processing the event, which includes executing some part of the application code. Eventually, the final actions associated to the event are committed and then the memory used is released. Simulation processes manage a future event list and processed event list and takes care of guarantee to emit events with the earliest time stamp. µsik supports multiple synchronization algorithms such as lookahead-based conservative protocol and rollback-based optimistic protocol (state-saving and reverse-computation). In an optimistic synchronization approach, μsik does state saving of the event processing during a time window, and commits all the events passing the earliest committable time stamp. Readers who are not familiar with parallel discrete-event simulation may find (Fujimoto, 2000) and (Perumalla, 2006) useful.

With Yades framework, users can run large demographic simulations that take advantage of High Performance Computing environments to run faster simulations. In this way, Yades handles the distribution of data and the work done by the parallel processing elements transparently. Moreover, it shows how transparency can be achieved in the context of socio-demographic fine-grained simulation models. With this approach, users do not have to account for parallel techniques for running demographic simulations thus making more transparent the use of parallel computing techniques. The model is specified through a user interface and the resulting source code is automatically generated by the framework, thus being much lighter that the corresponding parallel simulation code. Figure 3 shows the current approach followed by Yades in distributing the space and agents in a parallel architecture. Space is partitioned according to a logic entity (a region, a country, a neighborhood, a city, etc.). Agents located in that space are evaluated in the same processor where the region is computed. This schema relies on current demographic data bases and it is commonly used by other agent-based simulations (Collier and North, 2011; Rubio-Campillo, 2014). Therefore, users decide how many resources they need to request to run the simulation in parallel and how long the simulation will last. Yades requests this resources' allocation to the computer architecture where the code is executed (cluster or supercomputer). Furthermore, the approach of specifying agent-based simulation models in computer programs is not ideal for social scientists who are often not trained in coding. To solve this problem, we designed a web user interface for Yades' modeling and simulation. With the user interface modelers can define the set of agents, variables, and components that will be used in the simulation model. After specifying the model, they will be able to generate the simulation code and run it on the target execution platform such as a supercomputer, a cluster of PCs or even a local machine. As a result, modelers do not have to worry about the detail on how to harness the parallel computer power. The detailed implementation of Yades' physical and logical processes and the design of the web-user interface to approach social scientists and modelers can be seen in the Appendix.

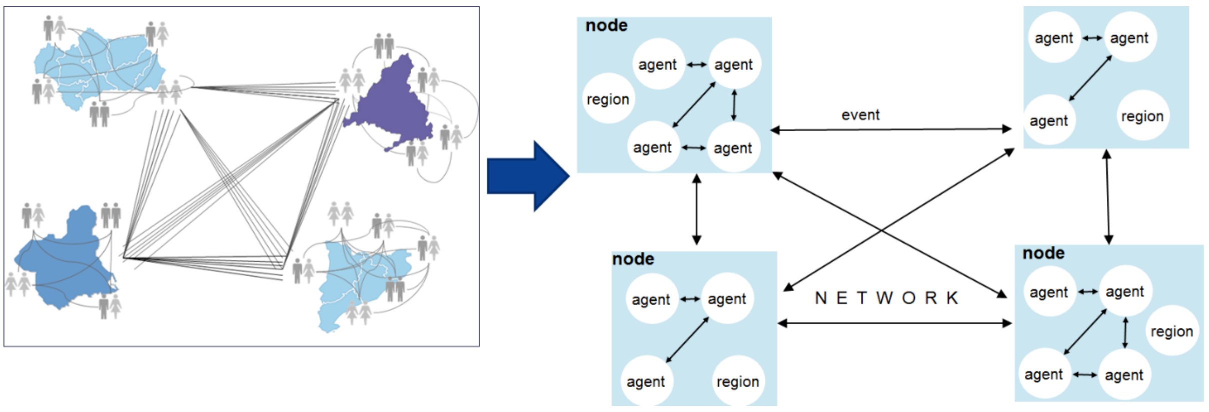


Figure 3. Schema on the translation between the logic model and the implementation in a parallel environment in Yades. Note that node here means processor.

5. Yades performance

To better understand the particular characteristics of social agent-based simulations and their performance limitations, we performed some experiments in Marenostrum, a supercomputer that is rank 77 in the TOP500 from June 2015. In this section, we present the results of experiments to understand Yades performance under varying conditions. For that, we use a synthetic population based on demographic data of the UK and run experiments to simulate its demographic evolution in terms of fertility, mortality, marital relationships, economic status, and migrations. The goal of the first experiment is to comprehend the effect of population size and migration activities on performance and scalability (Section 5.1 and Section 5.2). The objective of the second experiment suite is to study the performance of the tool on different execution settings: homogeneous environment (Section 5.3), heterogeneous population size (Section 5.4), heterogeneous processing speed (Section 5.5) and heterogeneous communication latencies (Section 5.6). The model uses a continuous time where future events can happen almost immediately. The lookahead is relatively small that makes a conservative protocol less efficient. For this reason, the optimistic protocol is used. All experiments were run using µsik settings that gave a roll-back based optimistic parallel simulation execution with a state-saving mechanism and a time window of 12 months (to limit how far a logical process can advance ahead of others). Fujimoto (2000, Chapters 4 and 5) provides a good overview of various techniques in optimistic parallel simulation.

The experiments were run on Marenostrum 2 and 3 supercomputers, with high-speed Myrinet and Infiniband interconnections respectively. Each node in Marenostrum 2 has two dual-core PowerPC 970 CPUs with a frequency of 2.3GHz and 8GB of memory. Each node in Marenostrum 3 has two 8-core Intel SandyBridge-EP E5-2670 with a frequency of 2.6GHz and 32GB of memory. In the experiments, we used up to 256 processors. While in Marenostrum 2 the program was compiled using gcc version 3.3.5 and mpich version 1.2.7 was used, in Marenostrum 3 we used gcc version 4.3.4 and OpenMPI 1.8.1. All performance results presented in this section are based on the average of five replications. Because the standard deviations are very low, we did not need more than five replications for each experiment. The results of the parallel simulation have been checked against the sequential execution for correctness.

5.1 Effect of population size on execution time and scalability

Perumalla (2005, 2007) has carried out a number of experiments to evaluate the performance of µsik simulation library. Hence, we do not repeat it in this paper. The focus of the following experiments is to understand the effect of population size on the overall simulation performance. We disable the migrations to measure the effect of the number of family units on computation time. The simulation for a period of 30 years was run with different initial population sizes of 80,000, 160,000 and 320,000 family units. The number of individuals is approximately twice the number of family units. Since the average fertility rate is set to be around two with no immigration, the numbers of individuals at the end of the simulation are approximately the same as their initial size. The result ran in Marenostrum 2 is shown in Figure 4. As we can see, it shows superlinear speedup, meaning the double of processors increases the speedup more than twice. For example, a simulation of 320,000 families in two nodes has a speedup of 4.66 while in four nodes it is 22.63, showing an improvement higher than two which would be expected in an ideal linear scalability.

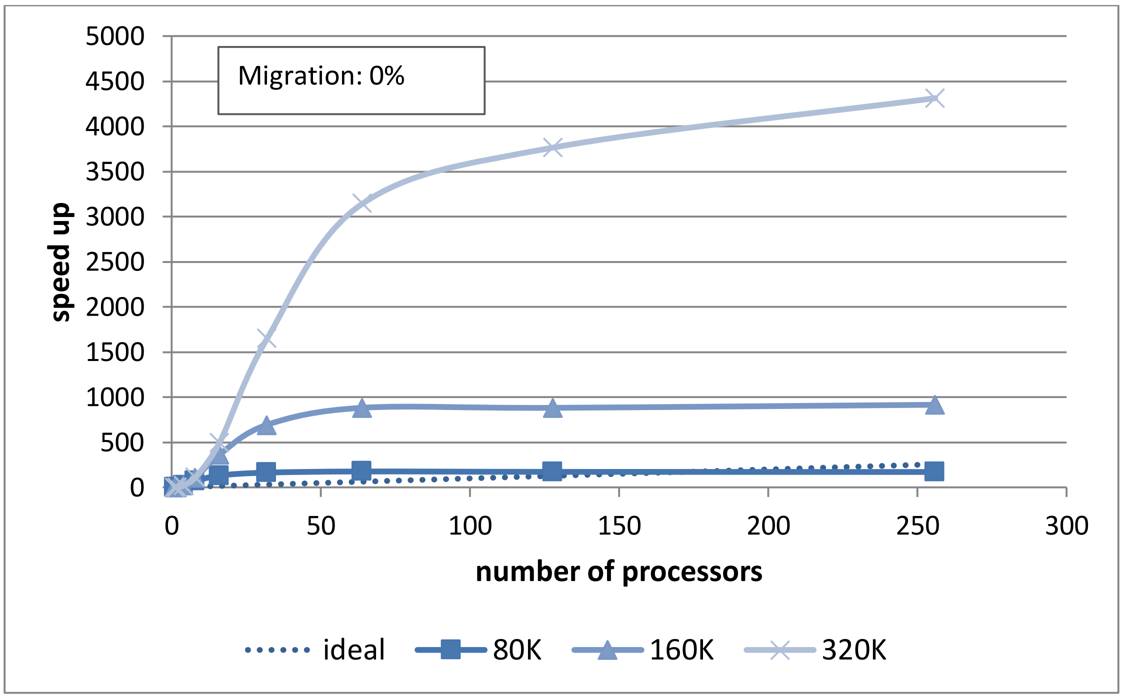


Figure 4. Effect of population size on speedup

The superscalar behavior can be explained due to cache memory problems. To illustrate that, we observe the behavior of cache miss ratio and IPC (instructions per cycle) in our application. Besides the studied population sizes, we run lower population sizes: 40,000, 20,000 and 10,000 family units. All the executions were made in one node. First, we plot the average miss ratio seen in L2 cache level in Figure 5. We choose L2 cache level because it is the most representative in Marenostrum 2. We observe average miss ratio increases with population growth, particularly when we go from 40,000 to 80,000 family units. Second, we plot IPC in Figure 6 observing a substantial decrease in IPC when we increase the number of families, especially beyond 40,000. These results prove our application is very sensitive to the size per node, since memory is determinant of performance which explains the super linear speedup found in our experiment.

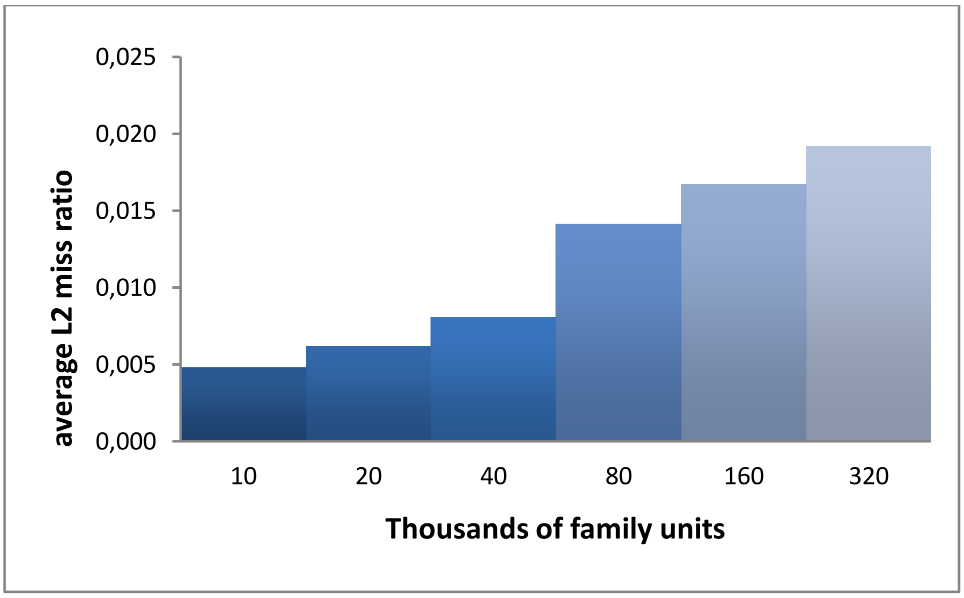


Figure 5. Miss ratio observed in L2 cache for different population sizes scenarios in one node

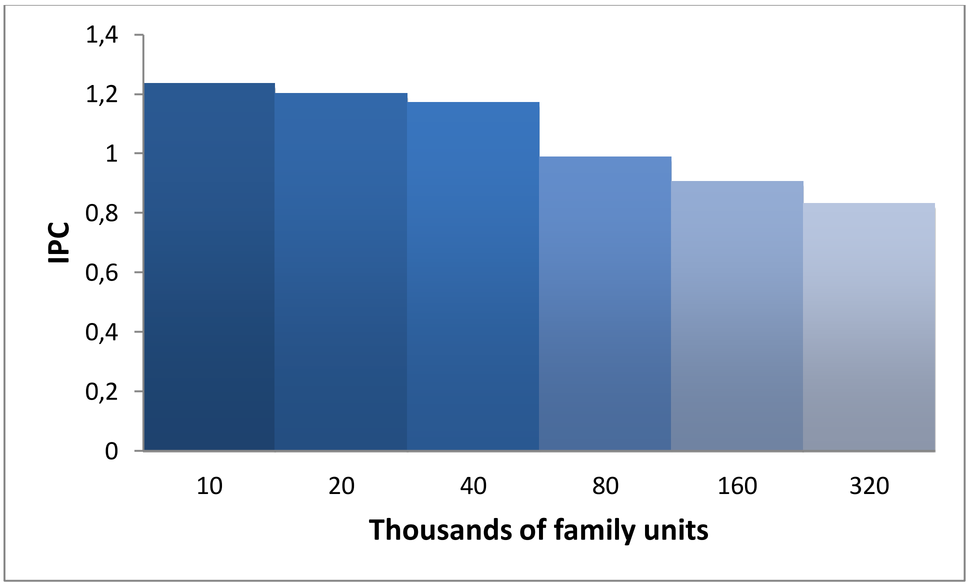


Figure 6. IPC obtained for different population size scenarios in one node

To deepen in the impact of different population sizes in performance, we calculate the effect of increasing the number of processors and the population size in proportion (weak scaling). We used three different population sizes configurations per node: 40,000, 20,000 and 10,000 family units. Due to Marenostrum 2 machine decommissioning, we perform the experiment in Marenostrum 3. Despite the difference on the architecture, the results (shown in Figure 7) are representative. We can see execution time remains stable when increasing the population and nodes. That means our application is scaling well as we increase the problem size when there are no migrations. Moreover, here we also observe the effect of cache misses on execution time. Looking at the results in one node, going from 20,000 to 40,000 population size the execution slows more than 74% (from 31.45 seconds to 2.02 minuts) in comparison to moving from 10,000 to 20,000 families (with a difference of 21.87 seconds).

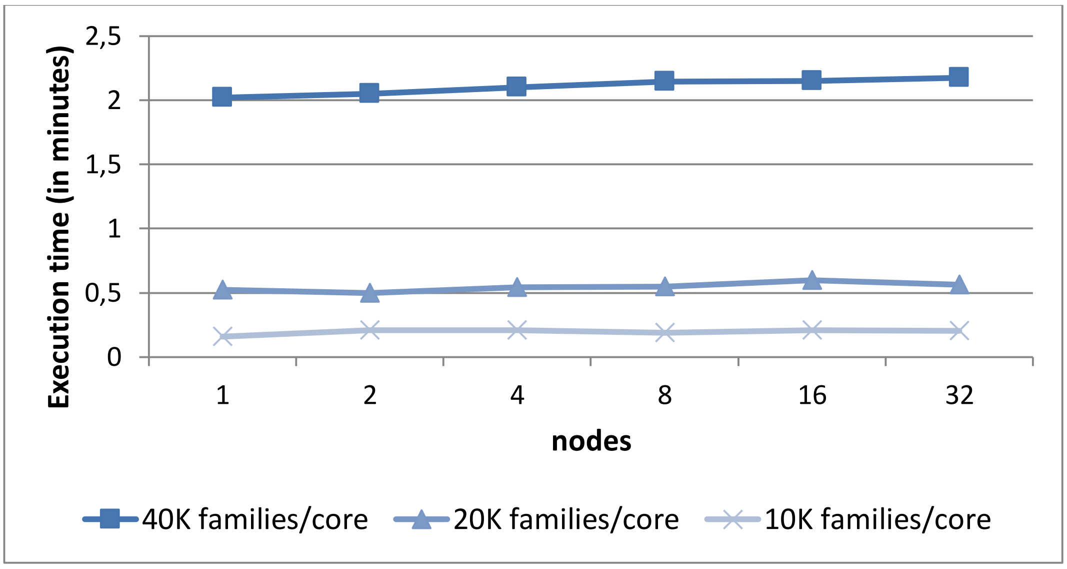


Figure 7. Weak scaling without migrations

A comparison between the simulation with different population sizes on Marenostrum 2 and 3 can be seen in Figure 8. The graph shows a comparative change in performance due to the difference on type of processors (from PowerPC 970 CPUs with a frequency of 2.3GHz to Intel SandyBridge-EP E5-2670 with a frequency of 2.6GHz), the change on memory per node (from 8GB to 32GB), and the presence of infiniband over myrinet.

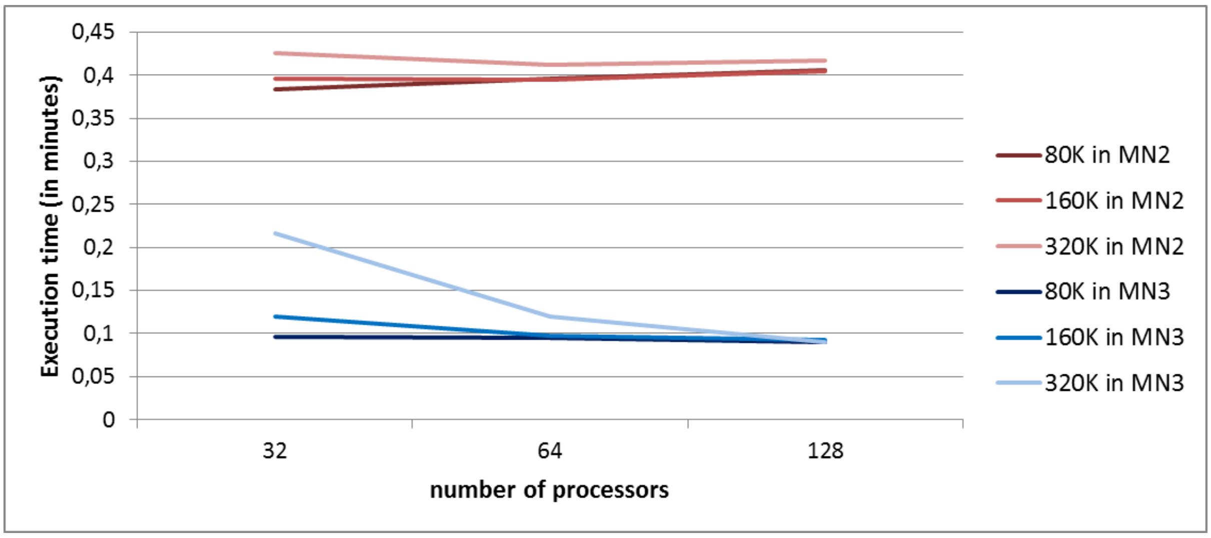


Figure 8. Comparison of simulation execution time between different population sizes (80,000 familie, 160,000 families and 320,000 families) in Marenostrum 2 (MN2) and Marenostrum 3 (MN3).

5.2 Effect of migrations on execution time and scalability

The objective of this experiment is to study the effect of varying the number of processors on execution time and speedup. Both experiments were run in Marenostrum 2.

In the first part, the effect of migrations on execution time is measured. The simulation was started with 320,000 family units and was run for a period of 30 years. We varied the number of processors from one to 256 using base-2 logarithmic scale. The probability of migrations was varied from 0% to 60%. The probability of migrations determines the probability of a family unit to migrate when there is a change in the employment status of one of the parents. The results are shown in Figure 9. On one hand, the execution time decreases as we increase the number of processors. Due to the increase cache misses and the IPC decrease shown in Figure 5 and Figure 6, the execution time is reduced more than half as we double the number of processors. For instance, from 2 to 4 processors we obtain a mean reduction of 79.41% without migrations and 68.83% with migrations (see Table **1**). Hence the simulation is slower for big scenarios in few processors due to the limitations on node memory. On the other hand, the reduction in the execution time becomes less significant as the number of processors increases. This is because the reduction in the computation cost becomes less significant and at the same time the communication costs becomes more expensive as the number of processors increases, due to migrations and rollbacks.

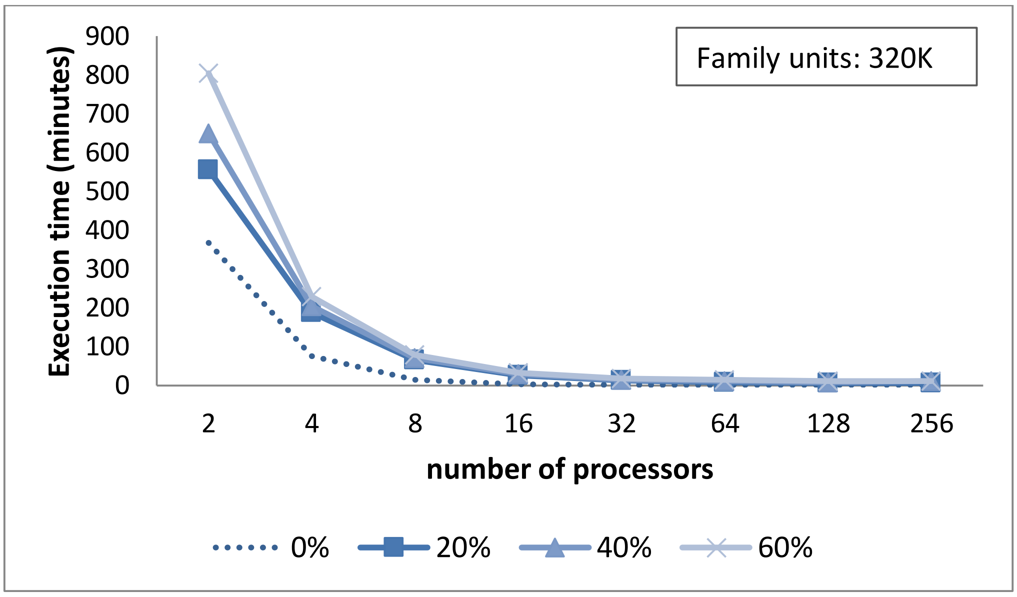


Figure 9. Effect of population degree of migration on execution time

Table 1. Percentages of time reduction as the number of processors increase under different migration scenarios

Increase in the number of resources	Migration probability			
	0	0.2	0.4	0.6
From 2 to 4 processors	79.42%	66,26%	68,67%	71,58%
From 4 to 8 processors	80.47%	65,24%	65,35%	65,49%
From 8 to 16 processors	76.74%	60,95%	59,32%	59,13%
From 16 to 32 processors	69.88%	52,11%	52,31%	46,83%
From 32 to 64 processors	47.46%	35,48%	21,30%	19,85%
From 64 to 128 processors	16.46%	26,89%	22,71%	22,39%
From 128 to 256 processors	12.68%	0,49%	3,93%	4,76%

In the second part, we study the speedup for a fixed population size (strong scaling). Figure 10 displays the performance improvement when the simulation is run for 30 years with 320,000 family units running on 1 to 128 processors. The speedups grow when the number of processors increases. Ideal line shows the linearity. The result shows the super linearity observed in section 6.1 due to cache misses. Moreover, for the same problem size, an increase in the number of processors increases computing power, but at the same time more synchronization overheads are required. This explains the diminishing performance gain as we increase the number of processors. It can also be seen in Figure 10 that the smaller the proportion of family units who are going to migrate, the larger the speedup obtained. This is because a high proportion of migrations increases the number of inter-processor communications when migrations take place and, consequently, the rollbacks. This also explains why the performance gain diminishes faster for the higher proportion of migrations.

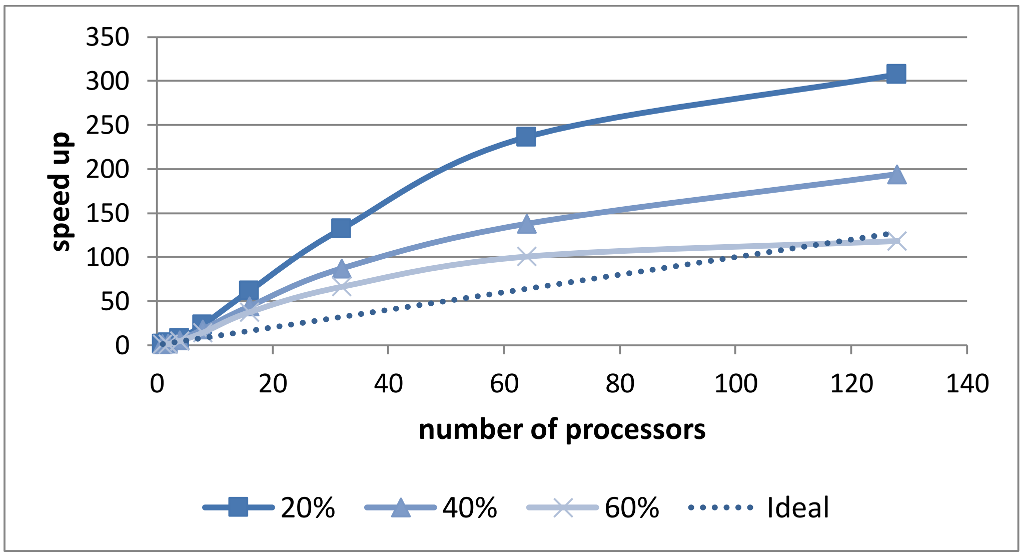


Figure 10. Strong scaling with migrations

5.3 Homogeneous environment

The objective of this experiment is to understand the effect of migration activities on the performance of the tool, specifically the execution time and the number of rollbacks, under an ideal execution configuration. In this configuration, we ran the simulation in Marenostrum 2 for a period of 30 years with an initial population size of 320,000 family units (around 650,000 individuals), divided equally among all administrative areas. This would produce a homogeneous workload to all processors. The simulation was run on one compute node containing four processors to minimize the effect of heterogeneous communication latency. The probability of migrations was varied between 0% and 60%. As explained earlier, migrations are responsible for all inter-processor communications in the simulation.

Table 2. Average number of migrations in 30 years

Probability of migrations	0%	20%	40%	60%
Average number of migrations (individuals)	0	324,801	642,065	946,524

The results are shown in Table 2 and Figure 11. As expected, the number of migrations is proportional to the migration probability (Table 2). Figure 11 shows that the increase in the number of migrations increases the execution time. The increase in the number of migrations increases the number of event that has to be executed by the simulator. As a result, it requires more time to execute all useful events. In this configuration (homogeneous environment), the average number of rollbacks is close to zero regardless of the migration probability. This indicates that each processor has enough computations and advances its simulation clock slower in such a way that the migrations seldom cause any rollbacks. Consequently, the overhead costs are mainly due to the inter-processor communications and rollbacks.

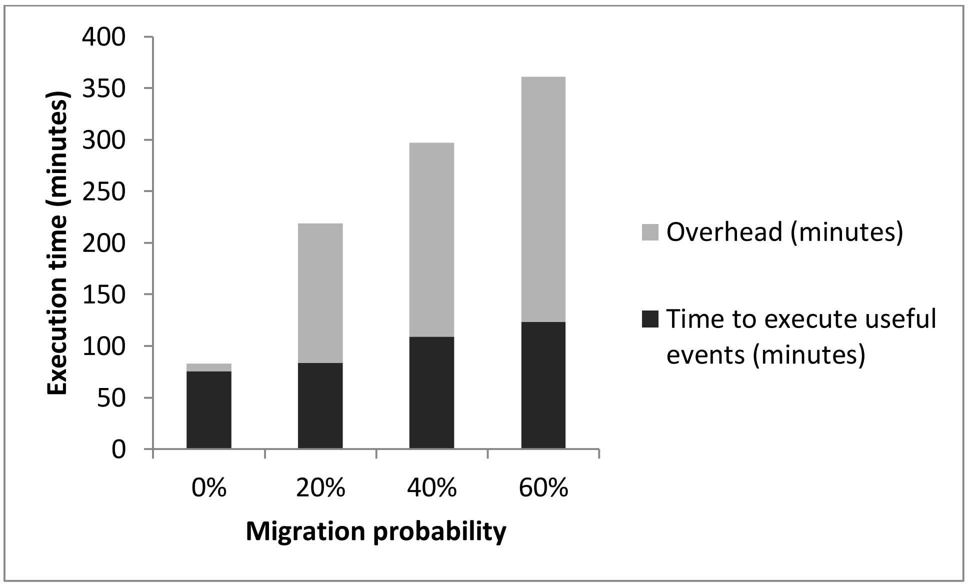


Figure 11. Execution Time for Homogeneous Workload

5.4 Irregular population size

In practice, the number of family units may vary across administrative areas. Hence, it is important to measure the effect of an irregular distribution of family units on the performance of the tool. As in previous experiments, we set the simulation duration to 30 years but fixed the probability of migrations to 60%. We ran the simulation on four processors in one compute node of Marenostrum 2 with a total of 320,000 family units at the start of the simulation. We varied the distributions of the family units from a regular distribution of 80,000 family units on each processing element to a highly irregular distribution of 215,000 family units on one processing element and 35,000 family units on each of the remaining processing elements (Table 3).

Table 3. Configurations of different population size accross processors

Configuration	Processor 1	Processor 2	Processor 3	Processor 4
Regular	80,000	80,000	80,000	80,000
Low irregularity	125,000	65,000	65,000	65,000
Medium irregularity	170,000	50,000	50,000	50,000
High irregularity	215,000	35,000	35,000	35,000

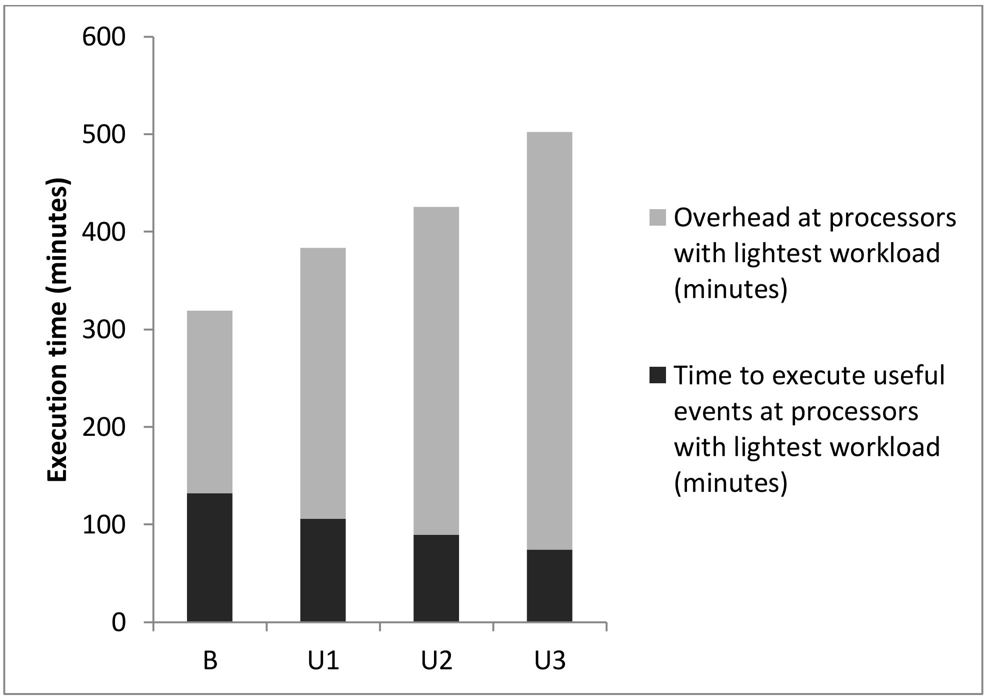
The four configurations are arranged in different columns in Table 4. Row 2 shows total number of migrations. As expected, the total number of migrations is roughly the same regardless of the distribution of the family units. Row 3 onwards shows the total number of rollbacks.

Table 4 .Effect of irregular distribution of Family Units on performance

Workload distribution	Regular	Low irregularity	Medium irregularity	High irregularity
Number of migrations (individuals)	946,524	947,359	946,532	948,299
Total rollbacks	0	252,397	348,495	474,626

Figure 12 shows that the equal distribution of family units across processors results in the best execution time. The worst execution time (almost two times slower) was given by the most irregular configuration in the experiments (configuration High irregularity). This result is consistent with what has been reported in parallel simulation literature, i.e. an equal distribution of family units will result in an equally distributed workload across the processors. Consequently, the processors can advance their simulation clock at a similar pace, which reduces the number of rollbacks.

A processor with the highest workload in the more irregular configuration has to execute more events. This explains the increase in the amount of time spent for executing useful events. In contrast, a processor with the lightest workload in the more irregular configuration will execute fewer events. Consequently, it spends less time executing useful events.



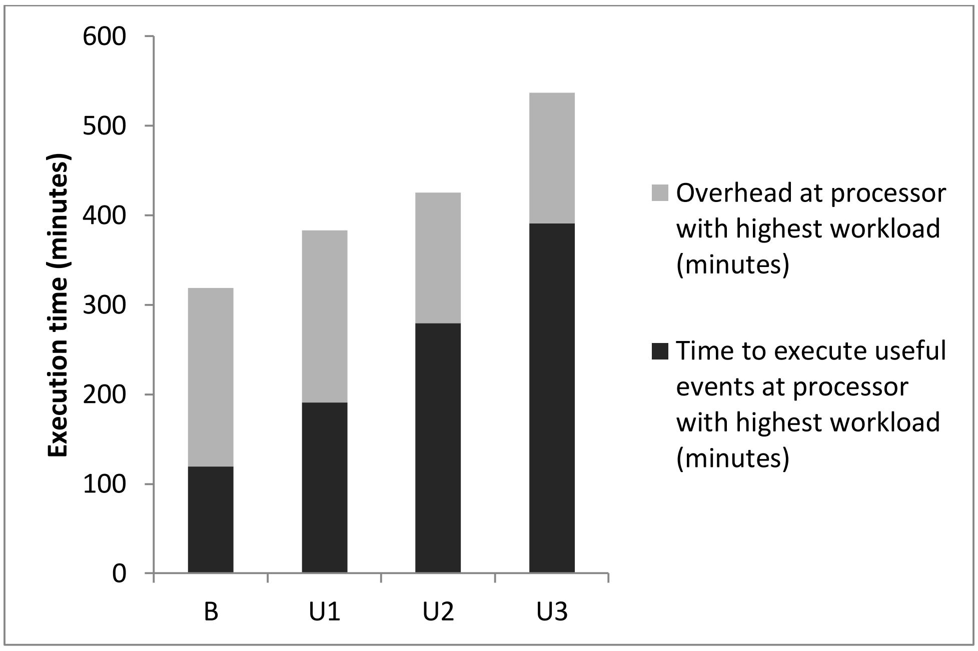


Figure 12. Effect of irregular distribution of Family Units on performance

In each of the irregular configurations, the processor with the higher workload will advance its simulation clock slower than the other processor; hence it will not experience any significant number of rollbacks. Consequently, its overhead can be attributed mainly to the communication costs other than rollback, such as waiting for events from another processor. The busier the processor, the less time is spent on waiting, which explains the decrease in the time spent for overhead. On the other hand, processors with lighter workload execute fewer events so they may advance their simulation time ahead of the busier processor. As a result, they have to rollback more often (see Table 3). This explains the increase in the time spent for overhead at the less busy processing elements.

5.5 Heterogeneous processing elements

In this section, we measure the effect of using heterogeneous processors on the performance of the tool. Heterogeneous environments are interesting for us since it is common for data centres to be often upgraded by replacing the part of the infrastructure with the latests processors and memories. Thus, the response of the framework to heterogeneous processing speed and heterogeneous communications can affect not only computation time but overheads.

In the experiment, we ran the simulation in Marenostrum 2 for a period of 30 years with an initial population size of 320,000 family units, divided equally among all administrative areas. The probability of migrations was fixed at 60%. To emulate the difference in processor speed, we inserted a delay for every simulation year at one of the processors (1 second and 2 seconds for each experiment, respectively). This is done by adding a delay to the event that generates an annual report. The result is shown in Table 5. The result is consistent with what has been reported in literature on parallel simulation, i.e., the wider gap in processor speed will result in more rollbacks (see the last row).

Table 5. Effect of irregularities in processor speed on performance

Delay (second)	0	1	2
Average number of migrations (individuals)	946,524	946,532	947,327
Average time to complete simulation (minutes)	361.1	429.1	480.2
Average number of rollbacks	0	252.4	348,495

5.6 Heterogeneous communication latency

Finally, we are also interested in the effect of heterogeneous latency in the communication between processing elements. The event size used in Yades is 512 bytes. For this event size, we used the Intel MPI Benchmark Suite to measure the inter-node latency and intra-node latency in Marenostrum 2 and found that the inter-node latency was 4 times slower than the intra-node latency. In the experiment, we used the same configuration as in the previous experiments but without any delay. We varied the locations of the four processing elements used in the experiment: using one compute node with four processing elements, using two compute nodes with two processing elements each, and using four compute nodes with one processing element each. The performance result is shown in Table 6. As expected, the number of migrations is about the same (row 2). The time spent in executing useful events is roughly the same because we expect similar number of useful events (row 4). The last two rows show that when the latency is homogeneous, the number of rollbacks is zero (row 6). As a result, it incurs some additional overhead cost (row 5). The overall performance (row 3) shows that a configuration with heterogeneous communication latencies (2×2) performs worse than a configuration with higher but more homogeneous communication latencies (4×1) due to rollbacks. However, the difference in performance is not very visible because the inter-node latency and intra-node latency are within the same order of magnitude.

Table 6. Effect of irregular communication latency on performance

Nodes × Processors	1×4	2×2	4×1
Average number of migrations (individuals)	947,327	946,990	949,679
Average time to complete simulation (minutes)	358	363.4	359.6
Average time to execute useful events (minutes)	124.1	124.2	122.7
Average overhead time (minutes)	233.9	239.2	238.7
Average number of rollbacks	0	6.4	0

6. Conclusion and future work

Because it is often difficult to study many social phenomena in laboratory situations, agent-based computational modeling provides a unique artificial laboratory to observe human behavior without the limits of empirical approaches (Billari et al., 2003). However, the increasing complexity of social models and the challenges of petascale era are making the use of parallel computing more necessary. Nevertheless, the parallelization of a simulation model requires making many critical decisions which have important impact on its performance and capability such as the number of cores required, or how the workload and data will be distributed and moved. Parallel simulation approaches for studying real social dynamics need to be studied and measured.

In this paper, we presented a solution to increase transparency of the use of parallel computing in performing demographic simulations. Our tool allows building agent-based demographic modelling and simulation to get a better understanding of large social dynamics. The main objective is to bridge the gap between the need for efficient parallelism exploitation, and the complexity of developing large complex population dynamics. For that, we run the social agent-based models on top of a parallel discrete-event simulation engine.

The main benefit of this outline is that our findings can be generalised to problems with similar characteristics to our demographic simulation model.

The tool allows modelers to specify individual behaviour such as fertility and change in marital status using agent-based simulation modelling paradigm and run the model on top of a parallel discrete-event simulation engine. We did not discuss the validation of the model because it has been presented somewhere else (Montañola-Sales et al., 2011). Our framework includes a user interface designed for social scientists who are not trained in parallel programming. Consultation with anthropologists during the design was fundamental to understand how our simulation user interface might be used by the end-users. Further work will include the evaluation of the web user interface by more invited end-users.

We also presented the performance evaluation result of Yades in a real High Performance Computing infrastructure. The performance measures such as speedup and execution time using up to 256 processors showed the potential of parallel simulation for large-scale scenarios. The application is sensible to the architecture where it is run in terms of memory consumption when the population size per node is high. However, when analyzing weak scaling we saw the application scales well when independent of cache issues. Moreover, migrations have a deep effect on performance due to their impact on network communications. We also conducted some more fine grained performance measures such as time spent in executing useful events, time spent for overhead and the number of rollbacks. Specifically, we have investigated the effect of three factors: irregular workload, heterogeneous processing speed and heterogeneous communication latency. The results are consistent with what has been reported in other application areas where parallel simulation has been used. Since the application of parallel simulation in demography is new, it is useful to quantify the effect of the three factors on performance. The findings are useful because it is likely that the simulation users will run the tool using heterogeneous population configurations.

We plan to add new functionalities such as allowing multiple administrative areas to be run on a processing element and introducing the concept of household which would allow one or more members of the same family unit to live in separate administrative areas, and to enrich the agent-based model to add macro variables as Human Development Index (DHI), Gross domestic product (GDP), and economic trends on regions to improve scenario simulations and test it with real case studies.

Acknowlegment

This research was supported by a PhD mobility grant from the Research and Innovation Ministry of the Spanish Government, by the Royal Society International Joint Project 2009/R2 grant number JP090402, and partially supported by the Secreteria d'Universitats i Recerca de la Generalitat de Catalunya under 2014 SGR 1534. We also want to thank Jordi Gassó to support the development of the GUI, inLab FIB from UPC-BarcelonaTech for hosting it, and Marta Garcia and Alessandro Pellegrini for their helpful comments on this manuscript.

References

Ahmad, S., & Billimek, J. (2005). Estimating the health impacts of tobacco harm reduction policies: a simulation modeling approach. *Risk Analysis*, *25*(4), 801–812.

Allen, T. T. (2011). *Introduction to Discrete Event Simulation and Agent-based Modeling*. Springer London. doi:10.1007/978-0-85729-139-4

Averill, M., & Kelton, W. D. (2007). *Simulation modeling and analysis Mcgraw-Hill series in industrial engineering and management science*. McGraw-Hill.

Bauer-Jr., D. W., Carothers, C. D., & Holder, A. (2009). Scalable time warp on blue gene supercomputers. In *Proceedings of the 23rd Workshop on Principles of Advanced and Distributed Simulation* (pp. 35–44). Picataway, N.J.: IEEE Computer Society Press.

Benenson, I., Omer, I., & Hatna, E. (2003). Agent-Based Modeling of Householders' Migration Behavior and Its Consequences. In F. C. Billari & A. Prskawetz (Eds.), *Agent-Based Computational Demography* (pp. 97–115). Heidelberg, Germany: Springer Physica-Verlag.

Billari, F. C., Ongaro, F., & Prskawetz, A. (2003). Introduction: agent-based computational demography. In F. C. Billari & A. Prskawetz (Eds.), *Agent-Based Computational Demography* (pp. 1–17). Heidelberg, Germany: Springer Physica-Verlag.

Bratman, M. E. (1999). Intention, plans, and practical reason.

Carothers, C. D., Bauer, D., & Pearce, S. (2002). ROSS: A high-performance, low-memory, modular Time Warp system. *Journal of Parallel and Distributed Computing*, *62*(11), 1648–1669.

Castle, C. J. E., & Crooks, A. T. (2006). Principles and concepts of agent-based modelling for developing geospatial simulations. *University College London, Working Paper Series, 110*. Retrieved from http://discovery.ucl.ac.uk/3342/

Coakley, S., Gheorghe, M., Holcombe, M., Chin, S., Worth, D., & Greenough, C. (2012). Exploitation of High Performance Computing in the FLAME Agent-Based Simulation

Framework. In *IEEE 14th International Conference on High Performance Computing and Communication 2012 (HPCC-ICESS)* (pp. 538–545). doi:10.1109/HPCC.2012.79

Collier, N. (2001). Repast: An extensible framework for agent simulation. *Natural Resources and Environmental Issues*, *8*(1), 4. Retrieved from http://digitalcommons.usu.edu/nrei/vol8/iss1/4

Collier, N., & North, M. (2011). Repast HPC: A Platform for Large-Scale Agent-Based Modeling. In *Large-Scale Computing* (pp. 81–109). John Wiley & Sons, Inc. doi:10.1002/9781118130506.ch5

Collier, N., & North, M. (2012). Parallel agent-based simulation with Repast for High Performance Computing. *SIMULATION*. doi:10.1177/0037549712462620

Cordasco, G. and De Chiara, R. and Scarano, V. and Carillo, M. and Mancuso, A. and Mazzeo, D. and Raia, F. and Serrapica, F. and Spagnuolo, C. and Vicidomini, L. (2011). D-Mason: Distributed Multi-Agent Based Simulations toolkit. Retrieved from https://sites.google.com/site/distributedmason/

Dahlen, F. J. (2009). LaMPsim - an open source microsimulation tool in development. In *Presented at the 2nd General Conference of the International Microsimulation Association, June 8th to 10th, 2009, Ottawa, Ontario, Canada*.

Eubank, S. (2002). Scalable, efficient epidemiological simulation. In *Proceedings of the 2002 ACM Symposium on Applied Computing* (pp. 139–145). New York, NY: ACM Press.

Ferrer, J., Prats, C., López, D., & Vives-Rego, J. (2009). Mathematical modelling methodologies in predictive food microbiology: a SWOT analysis. *International Journal of Food Microbiology*, *134*(1-2), 2–8. doi:10.1016/j.ijfoodmicro.2009.01.016

Forrester, J. (1971). *World Dynamics* (2nd ed.). New York, NY, USA: Productivity Press.

Fujimoto, R. M. (2000). *Parallel and distributed simulation systems*. Hoboken, New Jersey: John Wiley & Sons.

Galitz, W. O. (2007). *The essential guide to user interface design: an introduction to GUI design principles and techniques* (3rd ed.). Wiley & Sons.

Geard, N., McCaw, J. M., Dorin, A., Korb, K. B., & McVernon, J. (2013). Synthetic Population Dynamics: A Model of Household Demography. *Journal of Artificial Societies and Social Simulation*, *16*(1), 8. Retrieved from http://jasss.soc.surrey.ac.uk/16/1/8.html

Hammel E., M. C., & Wachter, C. (1990). *SOCSIM II, a sociodemographic microsimulation program, rev. 1.0, operating manual*.

Heiland, F. (2003). The collapse of the Berlin Wall: Simulating state-level East to West German migration patterns. In F. C. Billari & A. Prskawetz (Eds.), *Agent-Based Computational Demography* (pp. 73–96). Heidelberg, Germany: Springer Physica-Verlag.

Hinde, A. (1998). *Demographic methods*. London, UK: Arnold.

Hybinette, M., Kraemer, E., Xiong, Y., Matthews, G., & Ahmed, J. (2006). SASSY: A Design for a Scalable Agent- Based Simulation System Using a Distributed Discrete Event Infrastructure. In *Proceedings of the 36th Conference on Winter Simulation (Monterey,* *California)* (pp. 926–933).

Kniveton, D., Smith, C., & Wood, S. (2011). Agent-based model simulations of future changes in migration flows for Burkina Faso. *Global Environmental Change*, (0), -. doi:10.1016/j.gloenvcha.2011.09.006

Kokalj, A. (2003). Computer graphics and graphical user interfaces as tools in simulations of matter at the attomic scale. *Computational Material Science*, *28*(2), 155–168. doi:10.1016/s0927-0256(03)00104-6

Kurowski, K., de Back, W., Dubitzky, W., Gulyás, L., Kampis, G., Mamonski, M., … Swain, M. (2009). Complex system simulations with qoscosgrid. In *Computational Science--ICCS 2009* (pp. 387–396). Springer.

Lan, C., & Pidd, M. (2005). High performance simulation in quasi-continuous manufacturing plants. In M. E. Kuhl, N. M. Steiger, F. B. Armstrong, & J. A. Joines (Eds.), *Proceedings of the 2005 Winter Simulation Conference* (pp. 1367–1372). Picataway, NJ.: IEEE Computer Society Press.

Leombruni, R., & Richiardi, M. (2006). LABORsim: An Agent-Based Microsimulation of Labour Supply--An Application to Italy. *Computational Economics*, *27*(1), 63–88.

Li, N., Yi, W., Sun, M., & Gong, G. (2012). Development and application of intelligent system modeling and simulation platform. *Simulation Modelling Practice and Theory*, *29*, 149–162. doi:10.1016/j.simpat.2012.08.001

Lobb, C. J., Chao, Z., Fujimoto, R. M., & Potter, S. M. (2005). Parallel event-driven neural network simulations using the Hodgkin-Huxler neuron model. In *Proceedings of the 19th Workshop on Principles of Advanced and Distributed SImulation* (pp. 16–25). New York, NY: ACM Press.

Martin, D. E., McBrayer, T. J., & Wilsey, P. A. (1996). WARPED: A Time Warp Simulation Kernel for Analysis and Application Development. In *2013 46th Hawaii International Conference on System Sciences* (p. 383).

Mason. (n.d.). Retrieved from http://cs.gmu.edu/~eclab/projects/mason/

Meadows D.H., M. D. L. R. J., & William, W. B. (1972). *Limits to growth*. New York, NY, USA: Universe Books.

Meadows D.L., M. D. H., & Randers, J. (2004). *The limits to growth: The 30-year update*. London, UK: Earthscan.

Minar, N., Burkhart, R., Langton, C., & Askenazi, M. (1996). *The swarm simulation system: A toolkit for building multi-agent simulations*.

Mithen, S., & Reed, M. (2002). Stepping out: a computer simulation of hominid dispersal from Africa. *Journal of Human Evolution*, *43*(4), 433–462. doi:10.1006/jhev.2002.0584

Montañola-Sales, C., Onggo, B. S. S., & Casanovas-Garcia, J. (2011). Agent-based simulation validation: A case study in demographic simulation. In *SIMUL 2011, The Third International Conference on Advances in System Simulation* (pp. 101–107).

Montañola-Sales, C., Prats, C., Gilabert-navarro, J. F., López, D., Casanovas-Garcia, J., Valls, J., … Vilaplana, C. (2015). Modeling Tuberculosis in Barcelona. A solution to speed-up agent-based simulations. In L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, & M. D. Rossetti (Eds.), *Proceedings of the Winter Simulation Conference* (pp. 1295–1306).

O'Donoghue C., & Redway, H. (2009). Modelling migration in Pensim2: a dynamic microsimulation model of the British pension system. In *Presented at the 2nd General Conference of the International Microsimulation Association, June 8th to 10th, 2009, Ottawa, Ontario, Canada*.

Onggo, B. S. S. (2008). Parallel discrete-event simulation of population dynamics. In *Proceedings of the 40th Conference on Winter Simulation* (pp. 1047–1054). IEEE, Inc.

Onggo, B. S. S. (2010). Running Agent-Based Models on a Discrete-Event Simulator. In *Proceedings of the 24th European Simulation and Modelling Conference* (pp. 51–55). Hasselt, Belgium: Eurosis-ETI.

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

Pacheco, P. S. (1997). *Parallel programming with MPI*. San Francisco: Morgan Kaufmann Publishers Inc.

Parry, H. R., & Bithell, M. (2012). Large scale agent-based modelling: A review and guidelines for model scaling. In *Agent-based models of geographical systems* (pp. 271–308). Springer.

Pellegrini, A. (2015). *Techniques for Transparent Parallelization of Discrete Event Simulation Models*. Sapienza Università Editrice.

Pellegrini, A., & Quaglia, F. (2014). The ROme OpTimistic Simulator: A Tutorial. In *Euro-Par 2013: Parallel Processing Workshops* (pp. 501–512).

Pellegrini, A., Vitali, R., Peluso, S., & Quaglia, F. (2012). Transparent and efficient shared-state management for optimistic simulations on multi-core machines. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th International Symposium on* (pp. 134–141).

Perumalla, K. S. (2005). µsik: A Micro-Kernel for Parallel/Distributed Simulation Systems. In *PADS'05 Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation* (pp. 59–68).

Perumalla, K. S. (2006). Parallel and distributed simulation: traditional techniques and recent advances. In *Simulation Conference, 2006. WSC 06. Proceedings of the Winter* (pp. 84–95).

Perumalla, K. S. (2007). Scaling time warp-based discrete-event execution to 104 processors on a blue gene supercomputer. In *Proceedings of the 4th International Conference on Computing Frontiers* (pp. 69–76). New York, NY: ACM Press.

Prats, C., Montañola-Sales, C., Gilabert, J. F., Valls, J., Casanovas-Garcia, J., Vilaplana, C., … Lopez, D. (2016). Individual-based modeling of tuberculosis in a user-friendly interface: understanding the epidemiological role of population heterogeneity in a city. *Frontiers in Microbiology*, *6*(1564). doi:10.3389/fmicb.2015.01564

Rajaei, H. (1992). SIMA: an environment for parallel discrete-event simulation. In *Proceedings of the 25th annual symposium on Simulation* (pp. 147–155).

Rajaei, H., & Ayani, R. (1993). Design issues in parallel simulation languages. *Design & Test of Computers, IEEE*, *10*(4), 52–63.

Rao D.M., & Chernyakhovsky, A. (2008). Parallel simulation of the global epidemiology of avian influenza. In *Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation* (pp. 1583–1591). Picataway, N.J.: IEEE Publisher Press.

Rauner, M. S., Brailsford, S. C., & Flessa, S. (2005). Use of discrete-event simulation to evaluate strategies for the prevention of mother-to-child transmission of HIV in developing countries. *Journal of the Operational Research Society*, *56*(2), 222–233.

Read, D. (1998). Kinship based demographic simulation of societal processes. *Journal of Artificial Societies and Social Simulation*, *1*(1). Retrieved from http://jasss.soc.surrey.ac.uk/1/1/1.html

Richmond, P., Walker, D., Coakley, S., & Romano, D. (2010). High performance cellular level agent-based simulation with FLAME for the GPU. *Briefings in Bioinformatics*, *11*(3), 334–347. doi:10.1093/bib/bbp073

Rönngren, R., Liljenstam, M., Ayani, R., & Montagnat, J. (1996). Transparent incremental state saving in Time Warp parallel discrete event simulation. In *ACM SIGSIM Simulation Digest* (Vol. 26, pp. 70–77).

Rubio-Campillo, X. (2013). Pandora: An hpc agent-based modelling framework. Retrieved from https://github.com/xrubio/pandora/

Rubio-Campillo, X. (2014). Pandora: A Versatile Agent-Based Modelling Platform for Social Simulation. In *Proceedings of the SIMUL 2014 : The Sixth International Conference on Advances in System Simulation* (pp. 29–34). Nice, France: IRIA.

Rubio-Campillo, X., Cela, J. M., & Hernández-Cardona, F. X. (2012). Simulating archaeologists? Using agent-based modelling to improve battlefield excavations. *Journal of Archaeological Science*, *39*(2), 347–356. doi:10.1016/j.jas.2011.09.020

Salgado, M. (2013). The Evolution of Paternal Care. In A. Greenberg, W. Kennedy, & N. Bos (Eds.), *Social Computing, Behavioral-Cultural Modeling and Prediction* (Vol. 7812, pp. 1–10). Springer Berlin Heidelberg. doi:10.1007/978-3-642-37210-0_1

Saysel, A. K., Barlas, Y., & Yenigun, O. (2002). Environmental sustainability in an agricultural development project: a system dynamics approach. *Journal of Environmental Management*, *64*(3), 247–260.

Silverman, E., Bijak, J., Hilton, J., Cao, V. D., & Noble, J. (2013). When Demography Met Social Simulation: A Tale of Two Modelling Approaches. *Journal of Artificial Societies and Social Simulation*, *16*(4), 9. Retrieved from http://jasss.soc.surrey.ac.uk/16/4/9.html

Solcany, V., & Safarik, J. (2002). The lookahead in a user-transparent conservative parallel simulator. In *Proceedings of the sixteenth workshop on Parallel and distributed simulation* (pp. 11–16).

Tang, Y., Perumalla, K. S., Fujimoto, R. M., Karimabadi, H., Driscoll, J., & Omelchenko, Y. (2005). Optimistic parallel discrete event simulations of physical systems using reverse computation. In *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation* (pp. 26–35). New York, NY: ACM Press.

Tesfatsion, L. (2002). Agent-based computational economics: Growing economies from the bottom up. *Artificial Life*, *8*(1), 55–82. doi:10.1162/106454602753694765

Tisue, S., & Wilensky, U. (2004). NetLogo: A simple environment for modeling complexity. In *Proceedings of the International conference on Complex Systems, Boston, May 2004* (pp. 16–21).

Wu, B. M., & Birkin, M. H. (2012). Agent-based extensions to a spatial microsimulation model of demographic change. In *Agent-based models of geographical systems* (pp. 347–360). Springer.

Yoginath, S. B., & Perumalla, K. S. (2008). Parallel Vehicular Traffic Simulation using Reverse Computation-based Optimistic Execution. In *Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation* (pp. 145–152). Piscataway, NJ.

Zinn, S., Gampe, J., Himmelspach, J., & Uhrmacher, A. M. (2009). MIC-core: A tool for microsimulation. In *Proceedings of the 2009 Winter Simulation Conference, WSC 2009, Austin, TX, December 13-16, 2009. WSC 2009* (pp. 992–1002).

Appendix on Yades framework

1. Yades design and implementation

Figure 13 shows the Unified Modeling Language (UML) class diagram of the implementation of Yades model. To develop a simulation model in μsik parallel simulation library, we must specify three main components: a physical process (`PopulationSimulator`), a set of logical processes (`Region` and `FamilyUnit`), and a set of events (`PopulationEvent`). The following sub-sections explain how the three main components are implemented for the demographic simulation software.

1.1 Physical process

The physical process (PP), implemented as class `PopulationSimulator`, is defined as a subclass of class `Simulator`. The main tasks of this class are to: establish the simulation parameters, generate initial population, manage logical processes (agents) and generate simulation reports.

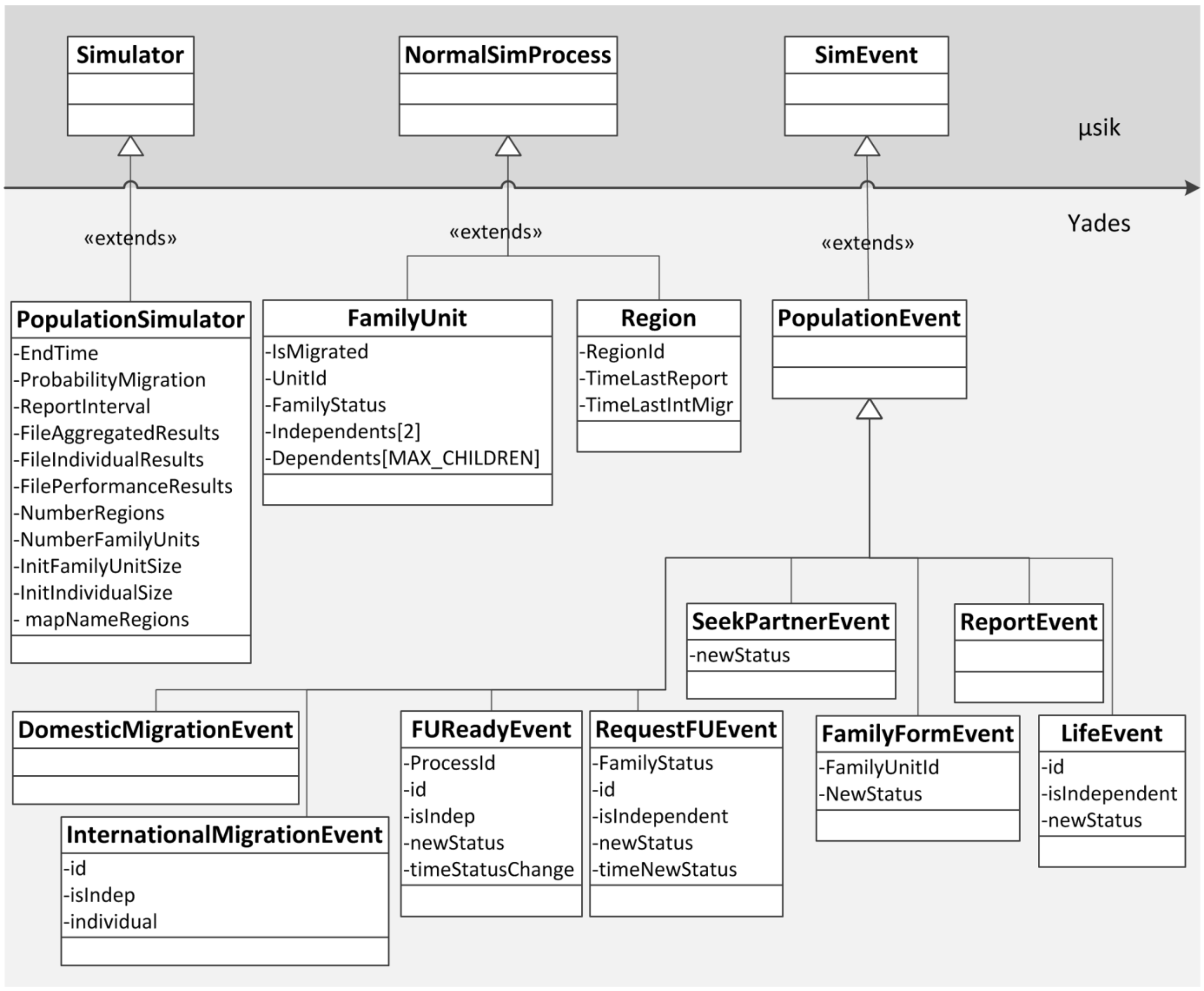


Figure 13. UML class diagram showing the classes and atributes of Yades simulation model and their inheritance from µsik library

Figure 14 shows that the simulator will initialize a number of physical processes, each of which will run on a processing element. Then, the initialization of two types of LPs: `Region` and `FamilyUnit` (FU) will follow. All logical processes Region represents an administrative area where a number of families live. Each logical process `FamilyUnit` represents a family in the community. Hence, for each physical process, there will only be one logical process `Region` and a number of logical processes for `FamilyUnit`. Communication between two logical processes occurs when a logical process (in this case a family) from the region in one physical process wants to migrate to another area on a different physical process.

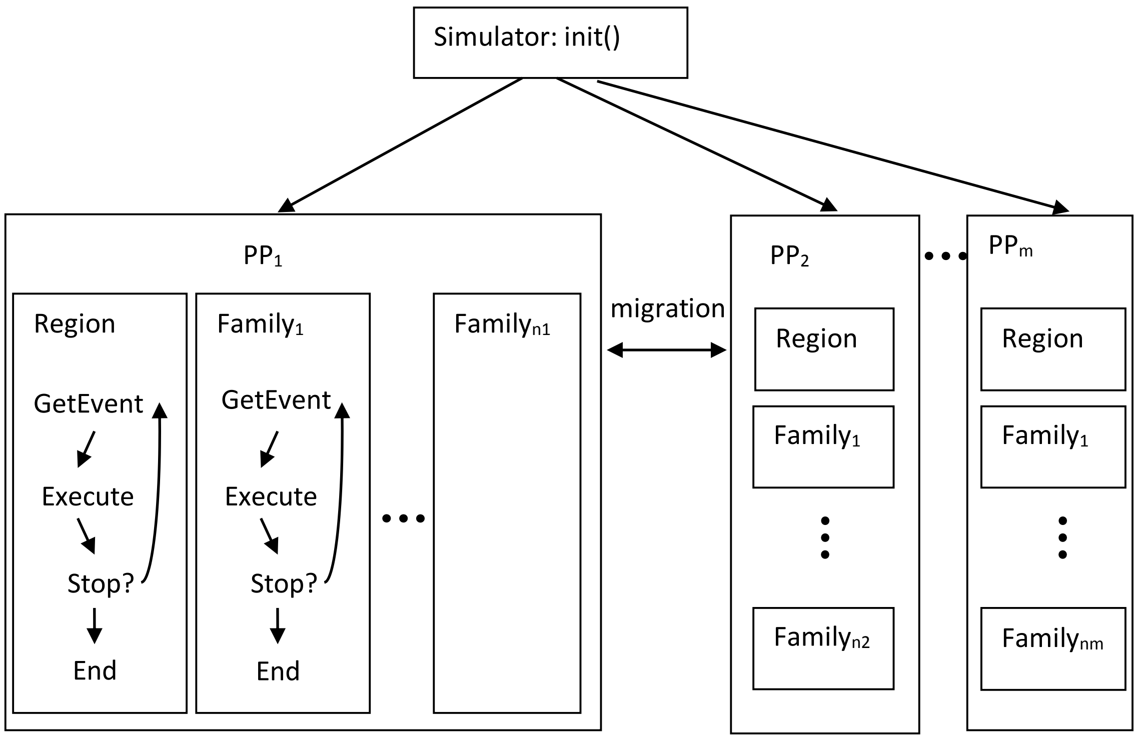


Figure 14. Structure and communication of the simulation, consisting on logical processes distributed in physical processes

One of the demographic modelers' main tasks is to provide data for the initial population in each region. The data includes the proportion of different age groups in the community, the proportion of different types of families by age group, proportion of different economic status by age group, proportion of different marital status by age group and the proportion of the number of children in a family.

1.2 Logical Process: Family Unit

One of the key design decisions concerns the types of logical processes that are going to be used in the design. We use family unit as one of the logical process types. The reason is that public policies may apply to individuals as well as groups of related individuals, such as households and single parents. For example, the UK Department for Work and Pensions and HM Revenue & Customs manage a number of public funds that may apply to individuals (including jobseeker's allowance and incapacity benefit) or groups of related individuals (that could include child benefit and housing benefit). Therefore, the model must recognize different types of 'policy unit.' Policy unit is often referred to as 'family unit.' A family unit is formed by either a single independent person or two independent individuals living together (as married, in civil-partnership, or in cohabitation) and any dependent individuals (children). Hence, in this definition, a family unit may represent an independent person, a single parent, a childless couple or a nuclear family. For completeness, the definition is extended to include orphans, that is, a family unit of dependent children without any parents. The decision to represent a family unit as a logical process has another benefit. When there is a change in the marital status that affects couples (such as from married to divorced or from married to widowed), only one message needs to be sent to the affected couple (in comparison to two messages, if a logical process (agent) was used to represent an individual).

A family unit may receive events that are related to five demographic components that may change the system states. Demographic modelers need to specify models for five demographic components: fertility, mortality, migration, a change in economic status and a change in marital status.

1.3 Logical Process: Region

The second logical process (agent) represents a region where a number of families live. This logical process will handle domestic migrations, immigration, changes in simulation parameters and periodic reports. Yades allows users to have regions with different population characteristics. The main limitation of the current version is that it only allows one processing element to run one region.

A region may receive a number of event types. The first event type is used when a family unit is going to migrate to a new area. This event will result in sending all members of the family to the new location. The second event type is used to simulate the immigration events. Demographic modelers need to implement models to represent immigration policies, such as the number and demographic characteristics of the immigrants. The third event type can be used by demographic modelers to specify periodical changes in simulation parameters such as life table and fertility rates. Finally, the report event can be used to produce periodical reports, for example, a report on the group structure (by gender, age group, marital status and economic status).

2. Yades Web user interface

Modelling and Simulation platforms are designed to support simulation modelling processes and help modelers to perform challenging models. Platforms are aimed to free modelers from the unnecessary part of model development processes that can be automated, accelerate the model development process and give a chance to reuse models and analyse results (Kokalj, 2003; Li et al., 2012). In the case of agent-based simulation, models have to be specified in computer programs/codes, usually using an object-oriented programming language. For social scientists, this approach is not ideal. To solve this problem, we present a web user interface as a framework for Yades' modeling and simulation that is available at http://yades.fib.upc.edu. The main advantage of using a web user interface is that it is easily accessible using any supported web browsers. The user interface is designed so that modelers can define the set of variables and components that will be used in the simulation model. The interface provides safety and authentication features through SSL connection and different types of role user since the model, the source code and the input/output data could be sensitive. It will also generate the simulation code so that users can download the code to be compiled and run on the target execution platform such as a supercomputer, a cluster of PCs or even a local machine. Therefore, modelers do not have to worry about the detail on how to harness the parallel computer power.

Figure 15 shows the sequence of the model implementation wizard that is designed to make the modelling specification process as easy as possible (the demographic modelers, of course, need to prepare the model specification off-line before they enter the specification to the web user interface). The sequence starts with the simulation configuration setting such as the simulation duration. Then, a number of geographical regions will be added. Next, modelers need to specify the initial group settings such as the gender proportion by age groups in each area. Then, the demographic components will be specified: fertility, mortality, marital status, economic status and migrations. To help defining qualitative and quantitative data, users can create distributions, regressions and logical rules in the system, which can be later use to define migration, fertility, economic or partnership changes. A detailed list of what the modelers need to specify for the model components is given in Table 6.

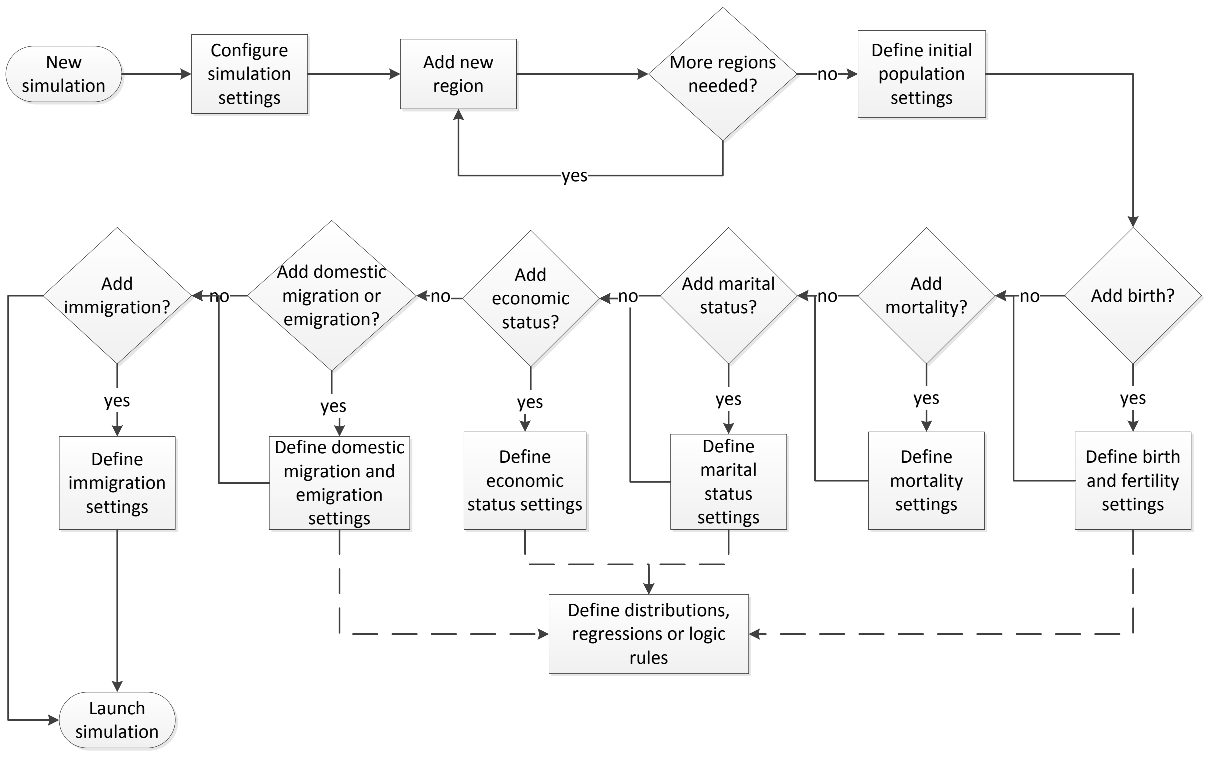
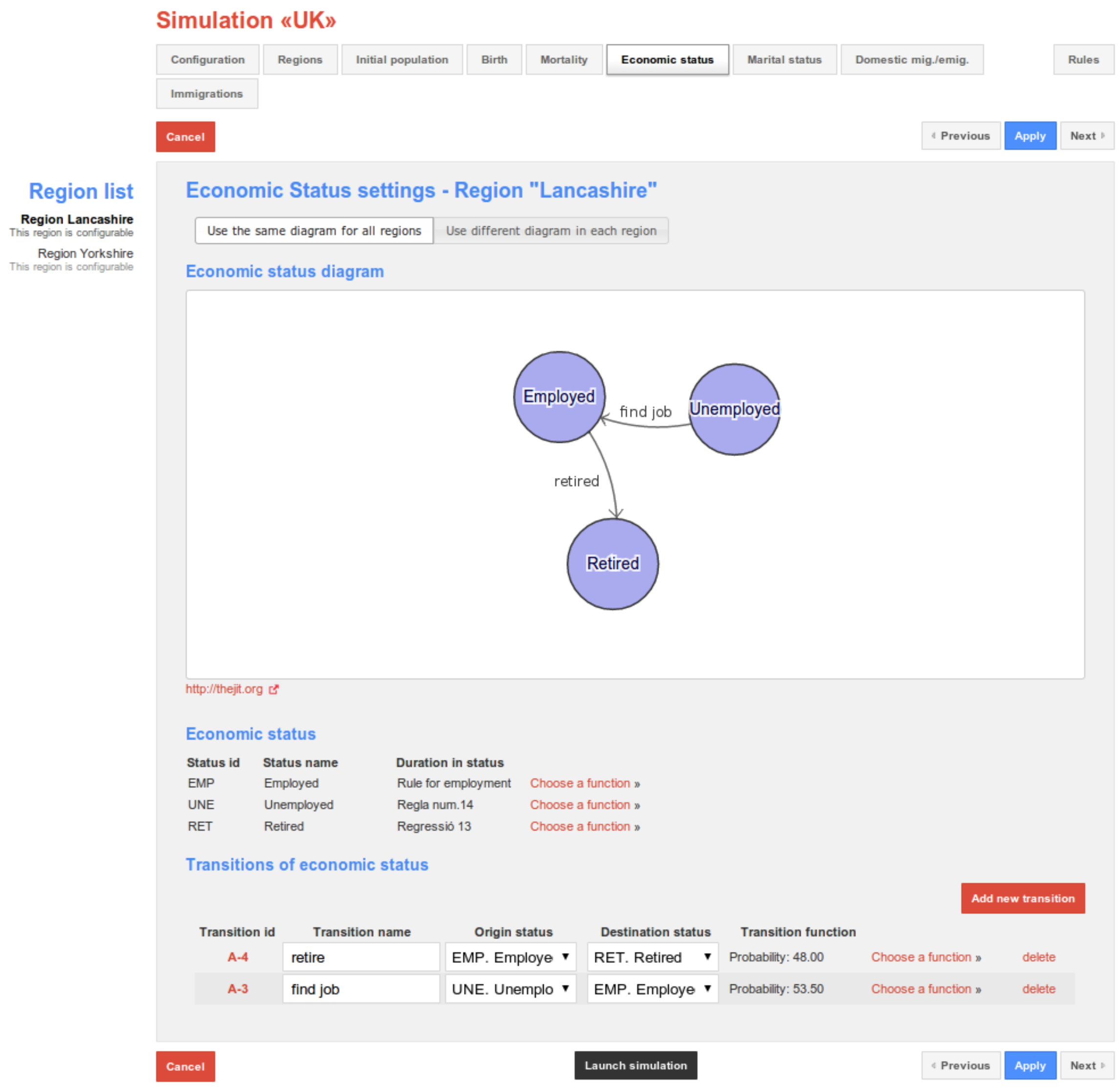

New simulation
Configure simulation settings
Add new region
More regions needed?
no
yes
Define initial population settings
Add birth?
no
yes
Define birth and fertility settings
Add mortality?
no
yes
Define mortality settings
Add marital status?
no
yes
Define marital status settings
Add economic status?
no
yes
Define economic status settings
Add domestic migration or emigration?
no
yes
Define domestic migration and emigration settings
Add immigration?
no
yes
Define immigration settings
Define distributions, regressions or logic rules
Launch simulation

Figure 15. Web user interface – GUI flow chart

The user interface is designed with the demographic modelers in mind. Typical demographic modelers (and many social scientists) are not trained in computer programming, let alone parallel programming. As a consequence, the web interface is designed to be user-friendly by providing both text and graphic development environment and following general user interface design principles (Galitz, 2007). The collaboration with anthropologists, who use demographic models, was fundamental to designing the user interface. Based on the discussion, Yades should provide functionalities for the users to represent their model using the commonly used formats such as regressions, logical rules, and standard theoretical distributions (see the examples in the list of parameters in Table 6). In addition, state-transition diagram is used to model changes in the marital and economic status (shown in Figure 16).

Table 7. Model definitions

Model component	**Description**	**List of parameters**
Configuration	Defines the simulation configuration	Simulation name, number of years, report interval, performance report, individual report, and age groups
Regions	Defines the regions in the model and their settings	Number of regions, homogeneous or heterogeneous setting, region name, family units per region, type of configuration
Initial population	Defines the initial population settings for each region	Population proportions by age group, types of family units by age group, economic status, marital status, children distribution, birthspacing distribution
Birth	Defines birth and fertility settings for each region	Fertility age interval, time to birth
Mortality	Defines mortality settings for each region	Life expectancy at birth or survival function or life table approach
Economic status	Defines economic status settings for each region	Duration in status, transition name, transition origin, transition destination, probability or rule for transition
Marital status	Defines marital status settings for each region	Duration in status, transition name, transition origin, transition destination, probability or rule for transition
Domestic migration/emigration	Defines the settings for national migration and emigrations	Logic function to determine whether the family unit is going to migrate, logic function to decide whether it is a domestic migration or emigration, migrations to a domestic destination, migrations to international destination
Immigration	Defines the settings of immigrant population	Number of monthly arrivals of immigrant family units, initial settings for immigrants
Rules	Defines distributions, regressions and logic rules that can be used in the simulation	Distribution name, distribution type, distribution parameters, regression name, regression parameters, rule name, type of result, rule sentence



Simulation «UK»

Region list

Region Lancashire
This region is configurable

Region Yorkshire
This region is configurable

Economic Status settings - Region "Lancashire"

Use the same diagram for all regions
Use different diagram in each region

Economic status diagram

Economic status

Status id	Status name	Duration in status	
EMP	Employed	Rule for employment	Choose a function »
UNE	Unemployed	Regla num.14	Choose a function »
RET	Retired	Regressió 13	Choose a function »

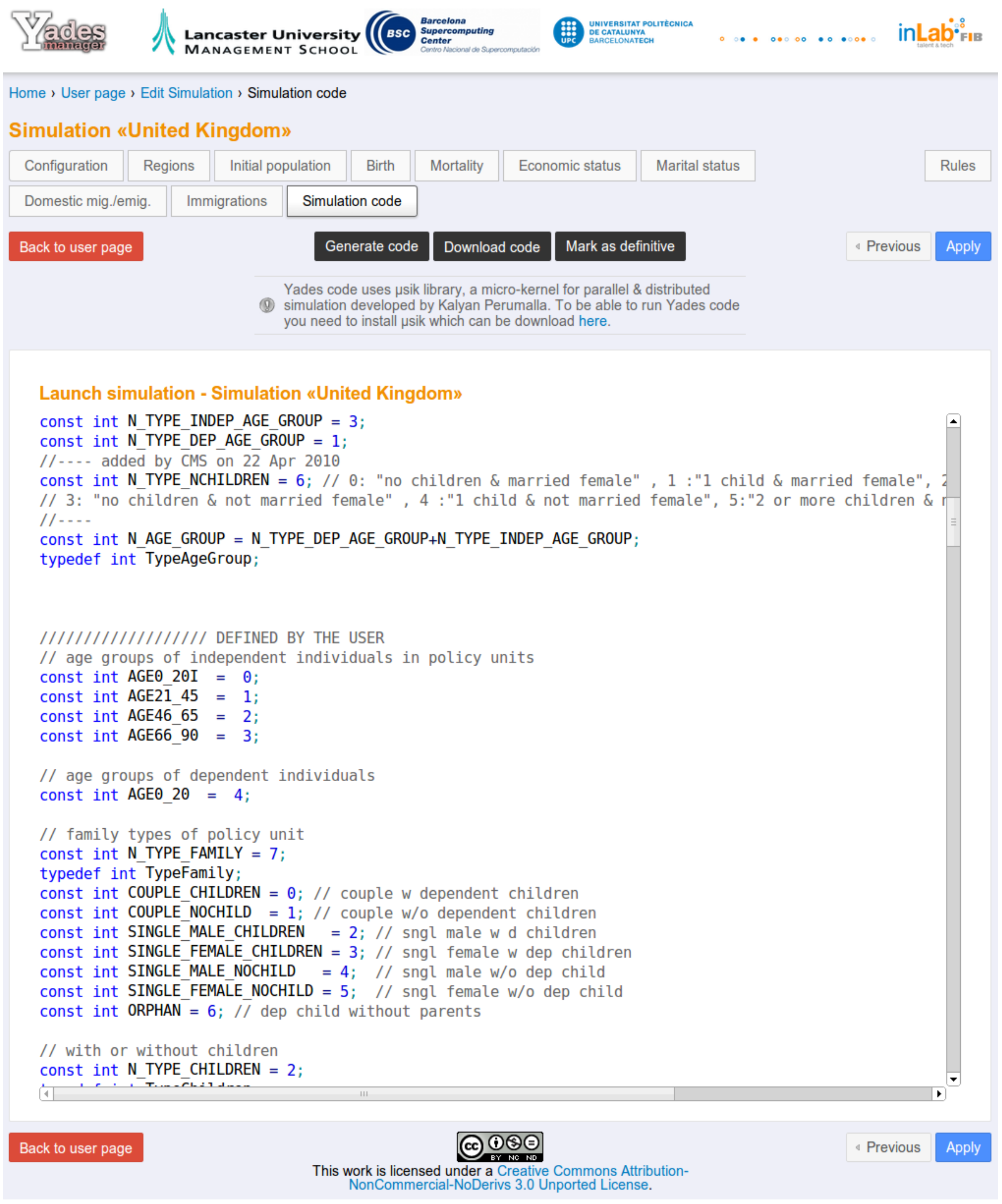
Transitions of economic status

Transition id	Transition name	Origin status	Destination status	Transition function		
A-4	retire	EMP. Employe	RET. Retired	Probability: 48.00	Choose a function »	delete
A-3	find job	UNE. Unemplo	EMP. Employe	Probability: 53.50	Choose a function »	delete

Figure 16. Web user interface – economic status model

Once the simulation model is defined users can launch the simulation. Then, the user interface shows the C++ code of the defined model as shown in Figure 17. The code can be compiled using a predetermined execution platform and it is portable to any High Performance Computing machine. In that way, users without developing and computing skills can execute it either in their desktop machines or submit it to a cluster of PCs or a supercomputing facility. Other agent-based simulation tools have similar functioning in leaving the management of job's execution to users, though there have been attempts to automatically generate parallel agent-based simulation code (Richmond et al., 2010). We have tested Yades using GNU C++ compiler 3.3.5 and 4.3.4, MPICH library 1.2.7 and

OpenMPI 1.8.1, libsynk communication library and µsik library. Both libsynk and µsik libraries are available from http://kalper.net/kp/software/musik/index.php.


Yades manager
Lancaster University Management School
BSC Barcelona Supercomputing Center Centro Nacional de Supercomputación
UPC Universitat Politècnica de Catalunya BarcelonaTech
inLab FIB

Home › User page › Edit Simulation › Simulation code

Simulation «United Kingdom»

Yades code uses µsik library, a micro-kernel for parallel & distributed simulation developed by Kalyan Perumalla. To be able to run Yades code you need to install µsik which can be download here.

Launch simulation - Simulation «United Kingdom»

```
const int N_TYPE_INDEP_AGE_GROUP = 3;
const int N_TYPE_DEP_AGE_GROUP = 1;
//---- added by CMS on 22 Apr 2010
const int N_TYPE_NCHILDREN = 6; // 0: "no children & married female" , 1 :"1 child & married female", 
// 3: "no children & not married female" , 4 :"1 child & not married female", 5:"2 or more children & 
//----
const int N_AGE_GROUP = N_TYPE_DEP_AGE_GROUP+N_TYPE_INDEP_AGE_GROUP;
typedef int TypeAgeGroup;



////////////////// DEFINED BY THE USER
// age groups of independent individuals in policy units
const int AGE0_20I  =  0;
const int AGE21_45  =  1;
const int AGE46_65  =  2;
const int AGE66_90  =  3;

// age groups of dependent individuals
const int AGE0_20  =  4;

// family types of policy unit
const int N_TYPE_FAMILY = 7;
typedef int TypeFamily;
const int COUPLE_CHILDREN = 0; // couple w dependent children
const int COUPLE_NOCHILD  = 1; // couple w/o dependent children
const int SINGLE_MALE_CHILDREN   = 2; // sngl male w d children
const int SINGLE_FEMALE_CHILDREN = 3; // sngl female w dep children
const int SINGLE_MALE_NOCHILD   = 4;  // sngl male w/o dep child
const int SINGLE_FEMALE_NOCHILD = 5;  // sngl female w/o dep child
const int ORPHAN = 6; // dep child without parents

// with or without children
const int N_TYPE_CHILDREN = 2;
```

Figure 17. Web user interface –simulation launched