doi 10.15826/recon.2015.2.016
UDC: 314.114:004.94

V. V. Okrepilov [a], ***V. L. Makarov*** [b], ***A. R. Bakhtizin*** [b], ***S. N. Kuzmina*** [a]
[a] State Regional Center for Standardization, Metrology and Testing in St. Petersburg and Leningrad region
[b] Central Economic Mathematical Institute of RAS

APPLICATION OF SUPERCOMPUTER TECHNOLOGIES FOR SIMULATION OF SOCIO-ECONOMIC SYSTEMS[1]

To date, an extensive experience has been accumulated in investigation of problems related to quality, assessment of management systems, modeling of economic system sustainability.

The studies performed have created a basis for formation of a new research area — Economics of Quality. Its tools allow to use opportunities of model simulation for construction of the mathematical models adequately reflecting the role of quality in natural, technical, social regularities of functioning of the complex socioeconomic systems.

Extensive application and development of models, and also system modeling with use of supercomputer technologies, on our deep belief, will bring the conducted researches of social and economic systems to essentially new level. Moreover, the current scientific research makes a significant contribution to model simulation of multi-agent social systems and that isn't less important, it belongs to the priority areas in development of science and technology in our country.

This article is devoted to the questions of supercomputer technologies application in public sciences, first of all, — regarding technical realization of the large-scale agent-focused models (AFM). The essence of this tool is that owing to increase in power of computers it became possible to describe the behavior of many separate fragments of a difficult system, as social and economic systems represent.

The article also deals with the experience of foreign scientists and practicians in launching the AFM on supercomputers, and also the example of AFM developed in CEMI RAS, stages and methods of effective calculating kernel display of multi-agent system on architecture of a modern supercomputer will be analyzed.

The experiments on the basis of model simulation on forecasting the population of St. Petersburg according to three scenarios as one of the major factors influencing the development of social and economic system and quality of life of the population are presented in the conclusion.

Keywords: Agent-based model, demographic projection, model simulation, quality of life, quality management methods, socioeconomic system, supercomputer technologies, scenario estimates, economics of quality

Agent-based models (ABMs) that have been actively developing recently are the most appropriate tool to simulate the behavior of complex socio-economic systems, as well as to evaluate multi-level quality management systems [1]. The basic idea behind the models of this class is to build a computing tool which is a combination of agents with a particular set of properties and which allows for the simulation of real phenomena. ABM differs from object-based models by the "activity" of its elements each having not only a given set of personal characteristics ("resources"), but also a target function ("interest") based on which its reaction to changes in the environment affecting areas of its interest (the "behavior") is simulated. The emergence of ABMs can be seen as the result of evolution in modeling methodology: the transition from mono-models (one model — one algorithm) to multi-models (one model — a set of independent algorithms). Thus, an ABM is an artificial society of autonomous interacting agents that can simulate a system as close to reality as possible. Agent-based approach to modeling is universal and easy to use for applied scientists due to its visual impact, but at the same time it imposes certain requirements on computing resources. It is obvious that direct modeling of lengthy social processes at national (or global) level generally requires considerable computational power.

In their turn, supercomputers allow for severalfold increase in the number of agents and other quantitative parameters (network nodes, territory size) in models originally developed for use on conventional desktop computers. Therefore, supercomputer simulation is a logical and desirable step for the simple models which have already been successfully tested on conventional computers. However, the specific architecture of modern computers does not guarantee that the computer model software will immediately work on a supercomputer. The computing core parallelisation and often its

[1] © Okrepilov V. V., Makarov V. L., Bakhtizin A. R., Kuzmina S. N. Text. 2015.

deep optimisation are required as otherwise the use of expensive supercomputer computation will not be justified.

In this regard, we should note that virtually all world congresses dedicated to ABM Development, one of the main issues in this area of knowledge, raise the problem of running ABMs on supercomputers.

Computer simulation is the broadest, most interesting and rapidly developing area of research that is in demand today in virtually all spheres of human activity. Agent-based modeling approach is universal and suitable for applied researchers and practitioners because of its visual impact, but on the other hand it imposes high requirements on computing resources. It is obvious that considerable computational power is needed for direct modeling of rather lengthy social process across the country and the world.

1. Foreign scientists and practicioners experience

Let us consider the most well-known examples of running agent-based models on supercomputers. In September 2006 a project was launched to develop a large-scale European economics ABM — EURACE, i.e. Europe ACE (Agent-based Computational Economics), with a large number of autonomous agents interacting within a socio-economic system [2]. The project involves economists and programmers from eight research centers in Italy, France, Germany, Great Britain and Turkey, as well as a consultant from Columbia University, USA — the Nobel Prize winner Joseph Stiglitz.

To overcome the limitations of widely used models, which consider aggregated agents assuming their rational behavior and balanced state, ACE methodology (Agent Based Computational Economics) was used as the basis for the study.

A geographic information system covering a wide range of facilities (companies, shops, schools, transport networks, etc.) is used for the model.

According to their developers, almost all existing ABMs consider either a separate branch, or a relatively small geographical area and, consequently, a small population of agents, whereas EURACE represents the entire European Union, so that the scale and complexity of this model is unique, and its numerical resolution requires the use of supercomputers, as well as special software.

To fill the model with statistical information the data (in the form of geo-information maps) of NUTS-2[2] level from the European Union statistical service that present information about 268 regions of 27 countries is used.

The model includes three types of agents: households (about 107), companies (about 105) and banks (about 102). They all have a geographical reference and are linked to each other through social networks, business relationships, etc.

EURACE is implemented using a flexible scalable environment for agent-based models simulation — FLAME (Flexible Large-scale Agent Modeling Environment) developed by Simon Coakley and Mike Holcombe, its initial purpose being to simulate the growth of cells under different conditions. The approach used in FLAME is based on so-called X-machines that resemble finite state machines, but differ from them in that each X-machine has a set of data (a kind of memory) and transitions between states are not only functions of the states, but also memory dataset functions.

So, each agent in the FLAME system is represented by an X-machine, with communication between the agents performed by means of sending messages. When working in parallel mode on a supercomputer, messaging between the agents requires extensive computational resources and due to that agents were initially distributed across the processors according to their geographical location. Thus, developers of the program minimised the computational load on the assumption that most of the communication between the agents takes place within a small social group, localised in a particular area. Hence, the entire model landscape was divided into small areas and distributed between the supercomputer nodes.

A series of experiments to study the labor market was conducted using the developed model. Without examining the numerical results obtained in detail, we should note that, in the authors' opinion, the main conclusion of the study is that two macro-regions with similar conditions (resources,

[2] NUTS (fr. Nomenclature des unités territoriales statistiques) — Nomenclature of Territorial Units for Statistics, which is a standard division of the country for statistical purposes, developed by the European Union and encompassing its member states. There are three levels of NUTS-units, the second level (NUTS-2) corresponding to administrative districts in Germany, counties in the United Kingdom, etc.

economic development, etc.) can display significant differences over a long period (10 years or more) due to the original agents' heterogeneity.

The EpiSims ABM developed by researchers at the Virginia Bioinformatics Institute considers both the agents' transitions and their contacts within the environment that is as close to reality as possible and contains roads, buildings and other infrastructure facilities. Building the model required a large amount of data, including information on the health of individuals, their age, income, ethnicity, etc.

The original purpose of the study was to build a large-dimension ABM to be run on a supercomputer which could be used to study how diseases spread in a community. However, later in the course of study other problems were also solved associated with the creation of specialised ABM ++ software which enables the development of ABM in the C ++ language and also contains functions that facilitate executable code distribution between the supercomputer cluster nodes. In addition, ABM ++ makes it possible to dynamically redistribute computational threads, as well as to synchronise events.

ABM ++, whose first version appeared in 2009, is the result of modernising the tools developed in 1990-2005 at Los Alamos National Laboratory in the course of building large-scale ABMs (EpiSims, TRANSIMS, MobiCom).

Inter-processor connections between computational nodes in ABMs often require synchronisation of events taking place in the model. ABM ++ allows for developing models that meet this requirement. For example, in social models agents often move between different points in space (work, home, etc.) which at program level corresponds to a cluster node change, and it is important for the receiving node model time to be synchronised with the time of the node that the agent has just left.

Also in ABM ++ the MPIToolbox library is implemented which connects the C ++ API (Application Programming Interface) interface and the Message Passing Interface (MPI) of the supercomputer and accelerates data transfer between the cluster nodes.

ABM ++ was created in Ubuntu Linux — the operating system with gcc / g ++ compilers. Eclipse package is recommended as an integrated development environment with plug-in to support C and C ++, as well as PTP (Parallel Tools Platform) plug-in to ensure the development and integration of applications for parallel computer architectures. Eclipse supports integration with TAU (Tuning and Analysis Utilities) Performance System — a tool for comprehensive analysis and debugging programs for parallel computing, which also facilitates agent-based models development.

Specialists from another research group at the same Virginia Bioinformatics Institute created a tool to study the spread of infectious diseases in different population groups — EpiFast, whose advantages include scalability and high execution speed. For example, simulating social activity of Greater Los Angeles (agglomeration with a population of over 17 million) inhabitants with 900 million connections between people on a cluster of 96 dual-core POWER5 processors took less than five minutes. Such high performance is provided by an original parallelising mechanism proposed by the authors [3].

As a rule, several natural ways are used to model a society on a group of processors. For example, one of them is based on the notion of a society as a set of unstructured connections between individuals. When parallelising, these connections are evenly divided into groups whose number corresponds to the number of processors. The disadvantage of this approach is the difficulty in tracking a particular individual's status — in other words, if, for example, an individual i is infected, the information about it should be synchronised in all groups, since we do not know which of them contains this person's connections. The need for such synchronisation entails a great processing load. Another approach is based on dividing people — model agents — according to their geographical location. However, in this case, since population is generally unevenly settled, processing load distribution requires the use of fairly complex algorithms for its alignment which also results in an additional computational load.

The method proposed by the authors is an even division of agents with their corresponding one-way outgoing connections into groups whose number is equal to the number of processors. The computational algorithm is based on the interaction of leading (master) processors and those being lead (slave processors) that is organised as follows: a leading processor "knows" which of the processors serves a particular agent, and each of the slave processors "knows" only the agents they are serving (local agents). During the computational process the slave processors send requests to leading processors about the connections with non-local agents. The leading processors do not perform any calculations, except searching for the non-local agent for each external connection, and forward the request to the appropriate processors.

According to the developers, the proposed algorithm allows for implementing effective high-speed simulations of systems with a large number of agents.

Classical models of epidemics spread are mainly based on the use of differential equations, but this tool makes it difficult to consider connections between individual agents and their many individual features. ABMs make it possible to overcome these disadvantages. In 1996 Joshua Epstein and Robert Axtell published a description of one of the first ABMs which examines the process of epidemics spread [4]. The model agents that differ in susceptibility to a disease, which depends on the immune system state, are distributed over a certain area. At the same time, in this model the agents, whose number is only a few thousand, display quite a primitive behavior.

Later one of the largest ABMs including data on the whole population of the United States, i.e., about 300 million agents, was built under the leadership of Joshua Epstein and John Parker in the Center on Social and Economic Dynamics at Brookings [5]. This model has several advantages. First, it allows for predicting the consequences of various types of diseases spread. Second, it is oriented towards supporting two calculation media: one medium consists of clusters with a 64-bit version of Linux installed, and the other comprises servers with quad-core processors and Windows OS installed (therefore, Java was selected as the programming language, although the developers do not specify which Java version was used). Third, the model is capable of maintaining agent numbers ranging from several hundred million to 6 billion.

The method of agents' distribution between hardware resources consists of two phases: first, the agents are distributed across computers involved in the operation, and then across threads on each computer. In course of model operation each thread may stop (at the time specified in advance) to send messages to other threads. All the messages prepared to be sent are stored in the message pool and then are dispatched simultaneously. Two auxiliary tools are also used in the model implementation: the first one controls threads on a separate computer, the second one ensures that all messages between threads have been sent before resuming the computational process.

When distributing agents across hardware resources two factors should be taken into account: 1) the node performance depends on the infected agents' number; 2) contacts which involve message transmission between threads require much more computational resources than those limited to local information. As a result, various agents' distributions are possible. On the one hand, it is possible to divide the entire geographical area being considered into equal parts whose number should correspond to the number of nodes, and then to assign a geographical region to each node. Such distribution allows for balancing the computational load between nodes. On the other hand, a specific territory representing a single administrative unit can be assigned to a particular node — in this case the processing load will be reduced as a result of reducing the number of contacts involving message transmission between threads. If the first method entails an increase in processing load due to resource-consuming contacts, the second method in some cases is fraught with significant imbalance between hardware resources. For example, the outbreak of any disease in one region will load one of the computing nodes, while the others will be idle.

The model being considered (US National Model) includes 300 million agents moving on the map of the country in accordance with the 4000\4000 dimension correspondence matrix specified by means of the gravity model. A computational experiment was carried out on the US National Model that simulates a 300-day spread process for the disease characterised by a 96-hour incubation period and a 48-hour infection period. The study discovered that the spread of the disease is on the decline after 65 % of the infected individuals have recovered and acquired immunity. This model is often used by specialists at Johns Hopkins University and the US Department of Homeland Security for research on strategies of quick response to different kinds of epidemics [6].

In 2009, a second version of US National Model was created, which includes 6.5 billion agents whose actions specification takes into account the available statistical data. It was used to simulate the impact of A (H1N1/09) flu virus spread on a global scale.

Earlier, a similar model was developed at the Los Alamos National Laboratory (USA) and its results were published on 10 April 2006 [7]. For the technical implementation of the model one of the most powerful supercomputers of the time — Pink — was used that consisted of 1024 nodes with two 2.4 GHz, 2 GB-memory processors. The scenarios of various viruses spread, including H5N1, were studied with this large-scale model comprising 281 million agents taking into account interventions of different kind: vaccination, closing schools and enforcing medical isolation in some areas.

Such tasks can obviously be performed not only by supercomputers, but also by cheaper solutions. One of such illustrative examples are scientists from The University of Newcastle, Australia, who built a cluster comprising 11 units — individual personal computers connected into a network with a capacity of up to 100 Mbit/s [8]. The software for cluster nodes was the same: the Debian GNU / Linux operating system and JavaParty — cross-platform software extension for Java language (JavaParty Software has the means for distributing code execution threads across cluster nodes). To evaluate the cluster performance three versions of one ABM were used to test various strategies of agents participating in the auction. The versions differed in the complexity of the agents' strategies.

The main conclusion of the test is that the use of clusters is justified only when the computational load of individual nodes is rather intense; otherwise, the threads distribution, on the contrary, reduces the model operation speed. We should note that such behavior of cluster applications is not specific.

Let us consider RepastHPC which is the development environment for high-performance computational agent-based models. The software developed to design ABMs for subsequent run on supercomputers — Repast for High Performance Computing (RepastHPC) — deserves special attention. This package is designed with C ++ and MPI — software interface to exchange messages between processes that perform tasks in parallel mode, as well as Boost library that expands C ++.

As part of RepastHPC, a dynamic discrete event planner to execute programming instructions with conservative synchronisation algorithms that allow for a delay of processes to ensure a certain sequence of their execution was developed.

In RepastHPC agents are distributed between processes, and each process is associated with an agent that is local to the process. In turn, the agent is local to the process executing the program code that describes this agent's behavior. Copies of the other, non- local agents may be present in any process, which allows agents from the entire model to interact with these copies. For example, let us suppose a user in their model involving parallel computation uses two processes — P1 and P2, each creating a certain number of agents and having its own scheduler to execute programming instructions. The agents whose behavior is calculated on P1 process are local to the process, and the program code can change their status only within the framework of this process (the same stands true for P2 process). Suppose P1 process requests a copy of A2 agent from P2 process. A2 agent is not local to P1 process, and, therefore, the program code executed in P1 process cannot change the A2 agent status. At the same time the agents processed as part of P1 process, if necessary, may request A2 agent's status, but the A2 agent's copy will remain unchanged. Changing the original A2 is only possible in P2 process, but in this case RepastHPC will synchronise agent status changes across all processes.

2. Adapting agent-based models for supercomputers: our approach

A model simulating Russian socio-economic system development over the next 50 years was run on "Lomonosov" supercomputer. This ABM is based on the interaction of 100 million agents that conventionally represent the socio-economic environment in Russia. The behavior of each agent is determined by a set of algorithms that describe its actions and interaction with other agents in the real world.

The project involved experts from Central Economics and Mathematics Institute of the Russian Academy of Sciences (V.L. Makarov, A.R. Bakhtizin) and Moscow State University (V.A. Vasenin, V.A. Roganov, I.A. Trifonov). The data for the simulation were provided by the Federal State Statistics Service and Russian Economic Situation and Public Health Monitoring Service. The model for a regular computer was converted into a supercomputer version [9]. Below we will consider the main stages and methods of effective reflection of the multi-agent system computational core on modern supercomputers architecture that we have worked out in the course of solving the corresponding problem.

Scaling problem. It is important to understand that scaling programmes for supercomputers is a fundamental problem. Although conventional and supercomputer programmes perform the same set of functions, the target functions they were developed for are, as a rule, different.

At initial stages of complex application software development designers try to reduce programming and staff training costs, improve transition between platforms, etc. in the first place, and leave optimisation for later stages. This is quite reasonable, since at early development stages functionality is a top priority.

However, after the start of the developed software implementation its capacity often turns out to be insufficient for processing large amounts of real data. Since modern supercomputers are not personal computers accelerated by a thousand times, a supercomputer program has to be significantly altered to be run on a supercomputer. Doing this effectively often requires special knowledge and skills.

If this has been properly done a significant increase in efficiency is usually achieved at three levels:

1) parallelising calculations;

2) specialisation of computing libraries according to tasks;

3) low-level optimisation.

Specialisation and low-level optimisation. Before seriously considering the use of supercomputers, the program should be optimised and adapted to the target hardware platform. Otherwise, the parallel version will only be a good test for the supercomputer, with calculations being inefficient.

Using a supercomputer without optimising and adapting the program to the target hardware platform is the same as sending a raw recruits regiment on a combat mission: they must first be well trained to perform their task (software specialisation and optimisation), as well as to effectively use weapons (low-level software optimisation) and only then resources will be used effectively.

Versatile simulation systems like AnyLogic provide universal procedures. A universal code can often be optimised for a specific group of problems.

Choosing modeling system support. Undoubtedly, ABMs can be programmed without a special environment, in any object-oriented language. The main disadvantage of the existing packages for ABM creation (except for RepastHPC) is that it is impossible to develop projects to be run on a computing cluster (i.e. no mechanism for parallelising program code execution is provided).

However, a more sensible approach would be to use a well-established system for ABMs — due to uniform implementation of typical methods for agents' interaction. Here we will only consider the ADEVS[3] system.

Adevs (A Discrete EVent System simulator) is a set of C++-based libraries for constructing discrete event simulations. Its advantages include:

— ease of implementation;

— high performance of models;

— support of basic numerical methods when building models;

— integrated simulation parallelisation using OpenMP;

— possibility to use standard parallelisation means;

— fairly rapid current development of libraries;

— cross-platform;

— low-level (current functionality does not impose any restrictions on the model);

— compiled code independence on non-standard libraries;

— open source code.

However, a significant disadvantage of this product is a complete lack of presentation tools and rather complicated model development in comparison with, for example, AnyLogic. Therefore, this product cannot be used to build models at customer level; however, it is an effective platform for parallel simulation.

The main elements of a program that uses the ADEVS library in constructing ABMs are usually as follows:

— ADEVS simulator:: Simulator <X>;

— ADEVS primitive agents:: Atomic <X>;

— ADEVS model (agents container):: Digraph <VALUE, PORT>.

Due to the above-mentioned advantages of the ADEVS system it was decided to implement the supercomputer version of the programme described below on its base. As part of this work MPI-version of the ADEVS simulator was developed, as well as visualisation system for the computational process on the basis of Qt library — a C++ cross-platform software development tool.

Next, we turn to a brief description of the developed model and procedures for its subsequent run on a supercomputer.

Initial agent-based model. The first phase of the described below ABM development is creating a tool to effectively solve the problem of the study on conventional computers, as well as setting model

[3] ADEVS system and its description. [Online resource]. URL: http://www.ornl.gov/~1qn/adevs, free. Title from the screen. (accessed on: May, 2014.)

parameters. After its successful testing with a small number of agents (about 20 thousand — this is the number of agents on which a high-performance computer is capable of performing calculations at a satisfactory rate, given the agents' complexity) it was decided to convert the model for a supercomputer — that was the second phase of development. The AnyLogic package was used at the first stage whose technical features allowed for quite fast model debugging and adjusting its parameters. Next, the model for a conventional computer was converted into a supercomputer version.

The image of the developed ABM working window is a picture that displays the agents as dots, a separate area in the upper right corner provides information on the number of agents as well as cartographic information and the selected region parameters, namely: name, population, GDP and other data the user sets for display. In the course of operation, the system can get operational information on the socio-economic situation in all regions of Russia, including the use of cartographic data that changes in real time depending on the values of endogenous parameters.

Model agents' specification was performed basing on the following key parameters: age; life expectancy; parents' specialisation; place of work; region of residence; income, etc.

Regions' specification was based on the following parameters: geographical boundaries; the number of inhabitants; the number of employees (by type); GRP; GDP per capita; investment volume; the volume of investment per capita; average wage; average life expectancy; population growth, and others. All these parameters were taken into account when testing methodological approaches to economic systems management [10], social patterns determination [11], as well as when developing the 2030 Strategy for the purpose of determining the values of the indicators affecting the main objective of the Strategy — the population quality of life.

To fill the model with data statistical reports from the Russian Federal State Statistics Service as well as RLMS sociological databases are used.

Converting models into a supercomputer programme. We have already discussed the problems associated with the use of ABM development tools for implementing projects that run on computational supercomputer clusters. In view of the difficulty of separating AnyLogic calculation part from the presentation part and due to the code implementation in a high-level Java language the code execution performance is significantly lower than with ADEVS, and moreover, it is very difficult (or very time-consuming) to process the generated code into a simultaneously executed programme.

The following is an algorithm for converting the AnyLogic model into a supercomputer programme.

The model translation. The models in the AnyLogic project are stored as XML-files containing the tree of the parameters necessary for code generation: agent classes, parameters, presentation elements, description of UML-diagrams of agents' behavior.

In the course of operation, this tree is translated into C ++ code of the programme that calculates the model. The tree is passed "in depth", with the following key stages singled out and aligned with the performance of the translation task.

1. Generation of the main parameters. Finding the root of the tree and reading the child nodes' parameters, such as the model name, assembly address, model type, type of presentation.

2. Generation of classes. Generation of classes (in more detail):

1) building a list of classes;
2) reading the class main parameters;
3) reading variables;
4) reading parameters;
5) reading functions;
6) generating a list of functions;
7) reading functions code;
8) function code conversion Java -> C++;
9) reading the figures used and controls;
10) generating shapes and controls initialisation code;
11) generating constructor, destructor, visualiser codes;
12) generating class structure;
13) generating header and source-files codes.

3. Simulator generation. Searching for the node which stores information about the simulation process (controls, important constants' values, elements of presentation, etc.).

4. Generation of common project files (main.cpp, mainwindow.h, mainwindow.cpp, etc.)

Importing input data. Data from the geographic information component of the original model (maps of Russia), containing all the necessary information, is uploaded into the model.

Generation of classes and conversion of functions code. When generating functions from the tree the following is read: function name, return type, parameters, and its body.

Based on the previously constructed class list, "heavy classes" in the arguments of functions, that is, all of the generated classes, classes of figures and other classes not included in the standard set, are replaced with the relevant indicators in order to save memory and avoid errors when working with it. Next, functions' headers are generated that are subsequently inserted into the header and source-files. With such reading the function body is converted from Java-code to a similar C ++ code by means of the appropriate function (Listing 5) (this is possible due to the rather narrow class of functions used, and in case of more complex functions manual completion of the translated code is required), after which it is added to the list of bodies for the class.

In the course of translation the task of converting the source code of the functions from the Java language into the C++ language repeatedly arises. It can be represented as a series of replacements of structures, such as the ones below.

— Converting cycles: Java-format.

— Converting indicators. In Java, there is no difference as explicit as in C ++ between an object and the object indicator, so the structure of work with them is the same. Therefore a list of classes is introduced, in which it is important to use operations with object indicators, not with the object itself, and which tracks all the variables of the classes, followed by the replacement within this function when addressing them in response to the respective address to indicators.

— Disclosure of "black boxes". In Java, and specifically in the AnyLogic library, there are a certain number of functions and classes, which have no clones in C ++, or in the ADEVS library. Therefore, additional shapes.h, mdb-work.h libraries were created, where the missing features are implemented.

— When generating the basic parameters of the classes list the names for the main class and simulated class-agents appear. The procedure for adding an agent into the simulator visibility range is added to the main class code.

Generation of external objects. In the process of generating external objects a separate function Main :: initShapes () is created, which contains all the "graphic information", that is, all shapes, whose classes are implemented in shapes.h, are initialized within the function. The corresponding example is shown in the next code fragment.

Generation of classes, header and source-files codes. The header and source-file of the corresponding class are created on the basis of the data read and generated.

Simulation generation. Pre-written files main.cpp, mainwindow.cpp, mainwindow.h, where templates specify the type of the main class and connected header files, were sufficient to generate the simulation. When compiling the source code the templates are replaced with the previously obtained (at the generation stage) class names. This is sufficient for dual-flow simulation, which can later be replaced by the appropriate module for multiprocessor simulation.

Additional parameters. At the tree parsing stage (see above) a tree that is similar in structure is formed to generate the C ++ code, which allows for setting the necessary compilation parameters (visualization of certain parts, visual validation of code recognition, additional assembly flags, etc.) at preparation stage.

Thereafter, when the transformation command based on these parameters is given the final compilation occurs.

Assembling the completed project. QtCreator, a freeware cross-platform integrated development environment to work with the Qt framework, is used to assemble the translated project.

Agent code. The source code was generated with the above-described compiler from AnyLogic (model.alp etc.) project files data (with the exception of the agent's behavior).

The behavior of the agent is to be generated from the status diagrams, but at the moment the automation of this process is not yet implemented. Thus, it was necessary to add a certain amount of code to the generated code.

After making the necessary changes a cross-platform application which repeats the main features of this model was developed as a result of compilation.

Statistics and visualisation of time sections. In view of non-interactivity in running the programme on large supercomputers output data collection and visualisation were separated (this is due to uneven

load on clusters at different times of the day with exclusive access not being possible). After the model recalculation the output data can be visualised again, taking into account the calculations and changes in the system made. Agents and certain regions can be displayed on the screen in different colours depending on the results obtained, which allows for visualising the dynamics of the changes that occurred in the modeling process.

Supercomputers available for calculations. At the time of the calculations three supercomputers

Table 1

Supercomputers available for the research group

Position on the Top-50 list	Supercomputers	CPU	Cores	TFlops
1	«Lomonosov» (Moscow State University)	12 422	82 468	902
3	MBC-100K (Joint Supercomputer Centre of RAS)	416	28 704	376
4	«Chebyshev» (Moscow State University)	256	6 400	320

were available (Table 1) included in the top five CIS Top-50 supercomputer ranking (version of 23.09.2014).

Two supercomputers were used in the study to perform calculations — « Lomonosov » and MBC-100K.

Results. The application of supercomputer technologies and program code optimisation made it possible to achieve high performance.

The code optimisation alone and the use of C ++ instead of Java allowed for the increase in the programme execution speed. Thus, the model was tested at the following initial conditions: 1) agents number — 20 thousand; 2) forecast period — 16 years (until 2030). According to the calculations results, it turned out that the model computation time when using ADEVS was 22 seconds on a single processor, whereas with AnyLogic it was 2 minutes 18 seconds on a single processor, which means that the development environment was well-chosen.

As noted above, a conventional high-performance personal computer is capable of calculating a set of about 20 thousand agents (the behavior of each of them is set by approximately 20 functions) at a satisfactory rate with the average time of model time unit (one year) conversion of about a minute. For a larger number of agents, for example 100 thousand, the computer simply “freezes”.

Engaging 800 supercomputer processors and implementing an optimised code allowed for increasing the number of agents to 5 million. Such an array of calculations was performed within a

Table 2

Calculations of St. Petersburg population under the three scenarios; thousands

Year	Baseline scenario	Wages decline	Migration police change
2015	5180,03	5150,02	5145,98
2016	5251,19	5148,02	5143,88
2017	5322,72	5144,17	5141,18
2018	5395,20	5133,57	5134,91
2019	5468,38	5133,23	5123,12
2020	5542,41	5125,41	5121,50
2021	5617,37	5122,19	5114,65
2022	5693,14	5120,09	5097,87
2023	5769,18	5101,64	5087,11
2024	5845,51	5094,12	5077,02
2025	5921,90	5092,93	5052,44
2026	5998,31	5083,76	5052,23
2027	6075,31	5056,77	5029,27
2028	6153,12	5041,34	5023,12
2029	6231,06	5034,31	5011,92
2030	6309,54	5023,77	4989,83

period of time approximately equal to 1 minute 12 seconds (this figure may vary depending on the type of processors being used).

Calculations for three scenarios were performed in the course of the study. The first one involved the current situation development (with unchanged values of the model parameters) — baseline scenario. The second scenario implied a wages decline in St. Petersburg by 15 % in relation to average wages in Russia. Thus, the model agents intending to migrate to Saint Petersburg from other regions in order to find better-paid jobs have lower motivation to move. The third scenario included the adoption of measures aimed at tightening migration policy, which also reduces the likelihood of agents moving from other regions. In the model it is represented by the decrease in the average values of probability parameters that determine the possibility of moving by 30 % of their initial values. The obtained results are shown in Table. 2.

The simulation results demonstrated that with current trends the population of St. Petersburg will have grown by almost 22 % by 2030. Meanwhile, much depends on economic conditions (in particular, the level of wages) and migration policy. The other two scenarios show population declines of 2.5 % and 3.1 %, respectively. In general, the latter scenarios seem highly unrealistic, but they too demonstrate an insignificant reduction in the number of St. Petersburg inhabitants, whereas the baseline scenario implies its considerable growth.

Thus, the research results can be used to improve the theory and practice of managing socio-economic development in regions, including the "North-West" macro-region, to enhance the effectiveness of management decisions, and to save all kinds of resources [12]. This, in turn, will increase the pace of the selected region's socio-economic development, enhance sustainability, reduce risks and improve quality of life indicators. The results obtained can also be applied in setting strategic directions for the development of a region as a socio-economic system, as well as in determining the prospects of such a development and its role in the overall economic development of the country.

References

1. Okrepilov V. V. (2013). Ekonomika kachestva kak metodologicheskaya osnova upravleniya regionami [Economy of ualityas methodological basis of management of regions]. *Ekonomika i upravleniya [Economics and management],* 1 (87), 8-14.

2. Deissenberg, C., Hoog, S. van der & Herbert, D. (2008, June). EURACE: A Massively Parallel Agent-Based Model of the European Economy. *Document de Travail,* 39.

3. Roberts, D. J., Simoni, D. A. & Eubank, S. (2007). *A National Scale Microsimulation of Disease Outbreaks.* RTI International. Research Triangle Park. Blacksburg: Virginia Bioinformatics Institute.

4. Bisset, K., Chen, J., Feng, X., Kumar, V. S. A. & Marathe, M. (2009). EpiFast: A fast algorithm for large scale realistic epidemic simulations on distributed memory systems. Yorktown Heights, New York; 2009:430–439. *Proceedings of 23rd ACM International Conference on Supercomputing (ICS'09).*

5. Epstein, J. M. (2009, August). Modeling to Contain Pandemics. *Nature, volume 460,* 687.

6. Collier, N. (2012, February 23). *Repast HPC Manual.* Available at: http://repast.sourceforge.net (date of access: 2013, May).

7. Keith, R. B., Jiangzhuo, C., Xizhou, F., Kumar, A. V. S. & Madhav, V. M. (2009). *EpiFast: A Fast Algorithm for Large Scale Realistic Epidemic Simulations on Distributed Memory Systems ICS'09.* June 8–12. N.Y.: Yorktown Heights.

8. Ambrosiano, N. (2006). Avian Flu Modeled on Supercomputer. *Los Alamos National Laboratory NewsLetter,* 7, 8, 32.

9. Makarov, V. L., Bakhtizin, A. R., Vasenin, V. A., Roganov, V. A. & Trifonov I. A. (2011). *Sredstva superkompyuternykh sistem dlya raboty s agent-orientirovannymi modelyami [Services of supercomputer systems for work with agent-focused models],* 3.

10. Babkin, A. V. & Shamin, L. K. (2008). Analiz primeneniya metodologicheskikh podkhodov k upravleniyu ekonomicheskimi sistemami [The methodological approache application analysis to economic system management]. *Nauchno-tekhnicheskie vedomosti SPbGU. Ekonomicheskie nauki [Scientific and Technical Journal of Peter the Great St. Petersburg Polytechnical University. Economic Sciences],* 1(53), 18-22.

11. Zusev, G. Yu. & Plotnikov, V. A. (2011). Sotsialnyye zakonomernosti i rol cheloveka v sovremennom ekonomicheskom razvitii [Social regularities and role of himan being in modern economic development]. *Nauchno-tekhnicheskie vedomosti Sankt-Peterburgskogo gosudarstvennogo politekhnicheskogo universiteta [Scientific and Technical Journal of Peter the Great St. Petersburg Polytechnical University. Economic Sciences],* 2, 119, 22-26.

12. Gnevko, V. (2012). Municipalities — roots of democracy and economics. *USA: Society and Science press,* 295.

13. Epstein, J. M. & Axtell, R. L. (1996). *Growing Artificial Societies: Social Science from the Bottom Up.* Ch. V. Cambridge, Massachusetts: MIT Press.

14. Parker, J. (2007). A Flexible, Large-Scale, Distributed Agent Based Epidemic Model. Center on Social and Economic Dynamics. *Working Paper,* 52, 25.

15. Lynar, T. M., Herbert, R. D. & Chivers, W. J. (2009). Implementing an Agent Based Auction Model on a Cluster of Reused Workstations. *International J. of Computer Applications in Technology,* 34, 4, 13-24.

Information about the authors

Okrepilov Vladimir Valentinovitch (Saint Petersburg, Russia) — Doctor of Economics, Professor, Member of the Russian Academy of Sciences, Honoured Worker of Science and Technology of Russia, Laureate of the State Prize of Russia, General Director of State Regional Center for Standardization, Metrology and Testing in St. Petersburg and Leningrad Region (1, Kurlandskaya st., 190103, St. Petersburg; e-mail: letter@rustest.spb.ru).

Makarov Valery Leonidovitch (Moscow, Russia) — Doctor of Physics and Mathematics, Professor, Member of the Russian Academy of Sciences, Director of the Central Economic Mathematical Institute of RAS (47, Nakhimovsky prospect, 47117418, Moscow; e-mail: makarov@cemi.rssi.ru).

Bakhtizin Alber Raufovich (Moscow, Russia) — Doctor of Economics, Leading Researcher at the Laboratory of Experimental Economics, Central Economic Mathematical Institute of RAS" (47, Nakhimovsky prospect, 117418 Moscow; e-mail: albert.bakhtizin@gmail.com).

Kuzmina Svetlana Nikolaevna (Saint Petersburg, Russia) — Doctor of Economics, Chief Specialist, State Regional Center for Standardization, Metrology and Testing in St. Petersburg and Leningrad Region (1, Kurlandskaya st., 190103, St. Petersburg; e-mail: kuzmina2003@bk.ru).