Development of Cloud-Based Microservices to Decision Support System

Konstantin Aksyonov, Olga Aksyonova, Anna Antonova(✉), Elena Aksyonova, and Polina Ziomkovskaya

Ural Federal University, 620000 Yekaterinburg, Russia
antonovaannas@gmail.com

Abstract. Intelligent systems of simulation become a key stage of the scheduling of companies and industries work. Most of the existing decision support systems are desktop software. Today there is a need to use durability, flexibility, availability and crossplatforming information technologies. The paper proposes the idea of working cloud based decision support system BPsim.Web and this one consists of some set of services and tools. The model of the multiagent resources conversion process is considered. The process of the simulation model developing via BPsim.Web is described. An example of the real process model is given.

Keywords: Simulation modeling · Intellectual systems · Artificial intelligence

1 Introduction

Creation of simulation systems (SIM) [1] is one of the promising directions for the development of decision-making systems for business processes (BP), supply chains and logistics [2, 3], technological processes (for example metallurgy [4]). Currently, the presence in SIM of communities of interaction agents that are identified with decision makers (DM) is significant [5–8].

Currently, commercial simulation solutions based on the market (such as AnyLogic, ARIS, G2) are desktop applications. ARIS system allows you to create html-pages with the results of experiments and upload them to the Internet. AnyLogic system is able to compile java applets with developed models and place them on the network. To start working with the model, it is necessary to fully download it to the user's electronic device, playing the simulation experiment of the model applet takes place on the user's device and requires significant computing resources.

The analysis showed that the greatest functionality of SIM of business processes is provided by AnyLogic and BPsim products. In the direction of service-oriented architecture, only G2 is developing. Thus, the urgent task is to choose a dynamic model of a business process and build on its basis a web-service of simulation. Comparative analysis of SIM is presented in Table 1.

Table 1. Comparative analysis of SIM.

№	Parameter	ARIS	G2	AnyLogic
1	Designing a conceptual domain model	No	No	No
2	Description language of BP			
2.1	Description of resources, mechanisms, converters	+	+	+
2.2	Hierarchical model of BP	+	+	+
3	Multi agent model	No	No	+
4	Simulation	+	+	+
5	Expert modeling	No	+	No
6	Web interface	No	No	No
7	Cloud computing	No	+	No
8	Non-programming user orientation	+	No	No
9	Existence of the free version	No	No	No

2 Features of Business Processes

From the point of view of the dynamic component of BP, the following basic requirements for models can be distinguished [2, 3, 5, 6]:

1) accounting for various types of resources [9, 10];
2) accounting for the status of operations and resources at specific times;
3) accounting for the conflicts on common resources and means [11, 12];
4) modeling of discrete processes;
5) accounting for complex resources (resource instances with properties, in the terminology of queuing systems - application (transaction));
6) application of a situational approach (the presence of a language for describing situations (a language for representing knowledge) and mechanisms for diagnosing situations and finding solutions (a logical inference mechanism according to the terminology of expert systems);
7) implementation of intelligent agents (DM models);
8) description of hierarchical processes.

3 Analysis of Existing Models Business Processes

Consider the following approaches and models of multi-agent systems and BP:

1) model of a multi-agent process of resource conversion;
2) SIE-model A.U. Filippovich;
3) models of active and passive converters (APC) B.I. Klebanov, I.M. Moskalev.

3.1 Model of Multi-agent Resource Conversion Processes

The dynamic model of multi-agent resource conversion processes (MARCP) [5, 6] is designed to model organizational and technical, BP and support of management decisions. The MARCP model was developed on the basis of the following mathematical schemes: Petri nets, queuing systems and system dynamics models. The key concept of the MARCP model is a resource converter having the following structure: input, start, conversion, control, and output. "Start-up" determines the moment the converter is started on the basis of: the state of the conversion process, input and output resources, control commands, means. At the time of launch, the conversion execution time is determined based on the parameters of the control command and available resource limitations. The MARCP model has a hierarchical structure. Agents manage the objects of the process based on the content of the knowledge base (KB).

3.2 Analysis of the SIE Model A.U. Filippovich

Integrated situational, simulation, expert model A.U. Filippovich (SIE-model) is presented in [13]. Due to the fact that this model is focused on the problematic area of prepress processes (printing), some of its fragments will be described in terms of the MARCP model. SIE-model is presented in the form of several different levels, corresponding to the imitation, expert and situational presentation of information [13]. The first level of the model is intended to describe the structure of the system. For this, a block is associated with each object (subject). Blocks are interconnected by interaction channels. Each block processes a transaction for a certain time and delays it for a time determined by the intensity. We formulate the following conclusions:

1. The SIE-model can serve as the basis for creating a multi-agent BP model.
2. The SIE-model has the following advantages: apparatus/ mechanism for diagnosing situations; a combination of simulation, expert and situational approaches.
3. The SIE model does not satisfy the requirements of the BP model: the presence of a DM (agent) model; problem orientation to business processes.

3.3 Model I.M. Moskalev, B.I. Klebanov

In the work of I.M. Moskalev, B.I. Klebanov [14–16] presents a mathematical model of resource conversion process, the specificity of which is the allocation of active and passive converters (APC). In this model, the vertices of graph X are formed by passive transducers, active transformers, stock of instruments and resource storages, and many arcs are represented by resource and information flows, flows to funds. The model of active and passive converters is focused on solving production-planning problems and based on scheduling theory. This model has not worked the possibility of implementing intelligent agents (models of DM) with a production knowledge base, as well as the implementation of a language for describing mechanisms and situations for diagnosing situations and finding solutions.

The results of the analysis of the considered approaches and models of dynamic modeling of situations are given in Table 2.

Table 2. Analysis of approaches and dynamic models of situations.

Characteristics	MARCP	SIE-model	Model of APC
1. Different types of resources	+	+	+
2. Time attendance	+	+	+
3. Conflicts on shared resources and tools	+	+	+
4. Discrete operation	+	+	+
5. Complex resource (application), application queue	+	+	+
6. Language for describing situations, diagnosing situations and finding solutions	+	+	No
7. Decision Maker Model	+	No	No

As follows from the table, all the requirements of the multi-agent business process model are met by the MARCP model. As the theoretical basis of the implemented method you can use the SIE-model, the advantage of which is the study of integration issues of simulation, expert and situational modeling.

4 Details of the Implementation of the MARCP Modification

To implement the simulation modeling service, it was decided to use the MARCP concept.

The simulation modeling service is based on ASP.NET Core technology in the C# programming language.

ASP.NET Core is a cross-platform, high-performance, open-source environment for building modern, cloud-based, Internet-connected applications.

ASP.NET Core provides the following benefits:

- A single solution for creating a web user interface and web API.
- Integration of modern client platforms and development workflows.
- Cloud-based configuration system based on the environment.
- Built-in dependency injection.
- Simplified, high-performance, modular HTTP request pipeline.
- Ability to host in IIS, Nginx, Apache, Docker or in your own process.
- Parallel version control of the application focused on .NET Core.
- Tools that simplify the process of modern web development.
- Ability to build and run on Windows, macOS and Linux.
- Open source and community oriented.

ASP.NET Core comes fully in NuGet packages. Using NuGet packages allows you to optimize applications to include only the necessary dependencies. ASP.NET Core 2.x applications targeting .NET Core require only one NuGet package. Due to the small size of the application's contact area, benefits such as higher security, minimal maintenance and improved performance are available.

Figure 1 shows the architecture of the simulation service.

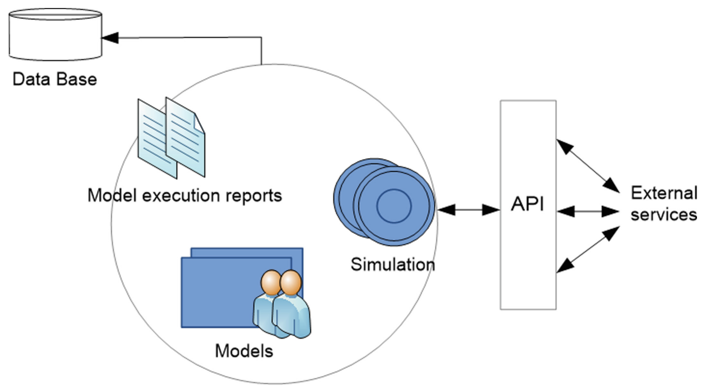


Fig. 1. Architecture of the simulation service.

The service manages 3 entities: models, simulation processes, and execution reports. It receives commands from the integration interface (API) and, depending on the command, receives or stores data in the database, performs internal transformations and calculations, and starts modeling.

5 WebAPI

The simulation modeling service has an interaction and integration interface in the form of cross-platform HTTP WebAPI. The HTTP protocol allows you to not only provide web pages. The simulation service describes a REST-style interface (RepresentationalStateTransfer). REST is an architectural style of software that defines the interaction of components of a distributed application on a network or the integration of multiple applications.

One of the requirements of REST is a unified programming interface for all entities with which the web service works. The CRUD (CreateReadUpdateDelete) interface of operations is described using the HTTP request verb (GET, POST, PUT, etc.), the names of entities and, if necessary, the identifiers of these entities.

Consider some possible queries on the models that the simulation service handles:

- GET model - get all models;
- GET model/{id} - get the model with the specified identifier;
- POST model - create a new model;
- PUT model/{id} - update the model;
- DELETE model/{id} - delete the model.

Here, all commands to the service have a single prefix “model”. If there is an appeal to many models at once - to get all models, add a new one to many models - only the prefix is used. If you need an action associated with a specific model - get, change, delete - you must continue the request URL with the model identifier. An HTTP verb is used to determine the action to be performed on the request object.

In the REST style, a special approach is laid down for requests that work with some service entities at once. This is how the imitation task is created:

- POST model/{id}/task.

The URL describes the relationship of one particular model with its tasks. A service should take one from the model domain with a specific identifier and refer to its many tasks for simulation. The HTTP verb indicates that a new task should be added to this set.

The receipt of all tasks for simulating a model is described in a similar way:

- GET model/{id}/task;
- GET task/{id}/report.

However, work with many tasks for imitation can be carried out not in the context of any particular model, but immediately with the whole set. The following methods are used for this:

- GET task - get all simulation tasks;
- GET task/{id} - get information on a simulation task;
- DELETE task/{id} - stop simulation.

For each task, many reports are generated. The task report is strongly related to the essence of the task itself. Therefore, deleting a task deletes all reports for this task. The reports do not have unique identifiers and can be received by the whole set.

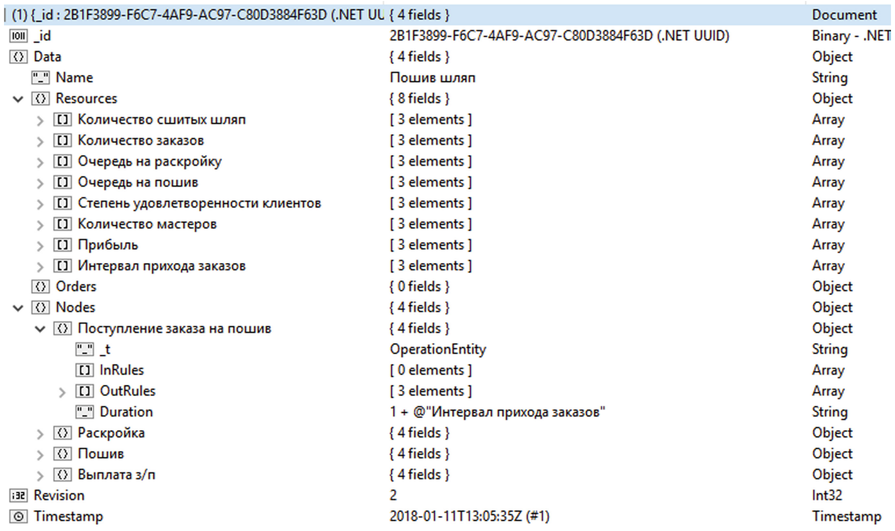
6 Data Storage

MongoDB was chosen as the data storage system of the SIM service.

MongoDB - DBMS that uses a document-oriented data model. This allows MongoDB to carry out CRUD operations very quickly, to be more prepared for changes in stored data and to be more understandable for the user. It is an open-source project that is written in C ++. All libraries and drivers for programming languages and platforms are also available in the public domain.

The storage method in MongoDB is similar to JSON (JavaScriptObjectNotation), although formally JSON is not used. MongoDB uses a format called BSON (binaryJSON) for storage. The BSON format has a certain structure and stores some additional information about a specific key and value pair (for example, data type and hash). Therefore, usually an object in BSON takes up a bit more space than JSON. However, BSON allows you to work with data faster: faster search and processing.

In MongoDB, each model is stored as a document. This document stores the entire structure of the model. Figure 2 shows the simulation model saved by the service in MongoDB.



```
(1) {_id : 2B1F3899-F6C7-4AF9-AC97-C80D3884F63D (.NET UL   { 4 fields }                                          Document
  _id                                                     2B1F3899-F6C7-4AF9-AC97-C80D3884F63D (.NET UUID)      Binary - .NET
  Data                                                    { 4 fields }                                          Object
    Name                                                  Пошив шляп                                            String
  v Resources                                             { 8 fields }                                          Object
    > Количество сшитых шляп                              [ 3 elements ]                                        Array
    > Количество заказов                                  [ 3 elements ]                                        Array
    > Очередь на раскройку                                [ 3 elements ]                                        Array
    > Очередь на пошив                                    [ 3 elements ]                                        Array
    > Степень удовлетворенности клиентов                  [ 3 elements ]                                        Array
    > Количество мастеров                                 [ 3 elements ]                                        Array
    > Прибыль                                             [ 3 elements ]                                        Array
    > Интервал прихода заказов                            [ 3 elements ]                                        Array
    Orders                                                { 0 fields }                                          Object
  v Nodes                                                 { 4 fields }                                          Object
    v Поступление заказа на пошив                         { 4 fields }                                          Object
        _t                                                OperationEntity                                       String
        InRules                                           [ 0 elements ]                                        Array
      > OutRules                                          [ 3 elements ]                                        Array
        Duration                                          1 + @"Интервал прихода заказов"                       String
    > Раскройка                                           { 4 fields }                                          Object
    > Пошив                                               { 4 fields }                                          Object
    > Выплата з/п                                         { 4 fields }                                          Object
  Revision                                                2                                                     Int32
  Timestamp                                               2018-01-11T13:05:35Z (#1)                             Timestamp
```

Fig. 2. Simulation model in MongoDB.

All models are saved in the Models collection. The structure of the model object itself, in addition to a unique identifier, consists of the following fields:

- Name - model name.
- Resources - an object in the form of a key-value, where key is the name of the resource, and value is the default value of the resource.
- Orders - the key-value object, where the key is the name of the request, and the value is an array of the names of the fields of the request.
- Nodes - a key-value object, where key is the name of the node, and value is a complex object that describes either the agent or the operation.

The "Agent" node has an array of global rules (GlobalRules) and a lot of knowledge (Knowledges). An array of rules stores the same rule entities as operations. Knowledge is a complex key-value object, where key is the name of knowledge, and value is an array of rules that describe this knowledge.

As mentioned earlier, simulation tasks also end up in the database. They are saved to the Tasks collection. The task consists of the following fields.

- ModelId - identifier of the model with which the task is associated.
- ModelRevision - version of the model by which the task was created. If the model has changed between the creation of the task and its immediate imitation, unforeseen incidents may occur.
- State - task state.
- SubjectArea - the subject area with which the simulation will occur.
- Configuration - the configuration with which the simulation will be launched. It stores information about when to stop the simulation, as well as some other information.
- Reports - an array of reports that were created for this task.

The task may be in one of the states. This is determined by the life cycle of the task. The states are as follows:

- 0 - Open - the task is created and ready to simulate;
- 1 - Scheduled - the task has already been planned, but has not yet been simulated (the engine is being prepared, objects are being created, etc.);
- 2 - Pending - the task is in simulation;
- 3 - Failed - the task failed;
- 4 - Successful - task completed successfully;
- 5 - Aborted - the task was interrupted;
- 6 - Deleted - the task was deleted before it was taken into imitation.

7 Simulation Process

The task created as a result of the HTTP request does not begin to be simulated immediately. It is saved to the database and enters the task queue. Service within a given period of time will check the queue for tasks. If there are no tasks for imitation, the cycle will be repeated after some time. If there is a task in the queue, imitation begins on it. The simulator can simulate several tasks in parallel. The degree of parallelism is not strictly defined and can be configured by the service settings. If the simulator does not have free resources to simulate, he does not look at the queue until they appear. As soon as some task completes the simulation, the simulator looks through the queue for new tasks.

When the task falls into the simulation, it saves to the database with the status 0 (Open). As soon as the simulator takes the task from the queue, it is set to status 1 (Scheduled). This is necessary so that a parallel process or another instance of the service does not begin to simulate the same task. When the simulator finishes preparations, he will begin the simulation, setting the task status 2 (Pending). Upon successful completion of the task, it will receive the status 4 (Successful), and if an error occurs - 3 (Failed). The task can be canceled by a special team. In this case, she will have the status 5 (Aborted) or 6 (Delete).

Figure 3 shows the pattern of product movement between the hot rolling mill LPC-10 and the cold rolling mill LPC-5.

The task is to recreate the production process of sheet steel coils and conduct a series of studies in which it is necessary to evaluate a set of key parameters within three 24-h working days. The parameters are as follows:

1. The minimum number of slabs in the warehouse of cast slabs at the beginning of the simulation, providing a continuous supply of slabs every three minutes.
2. The current, minimum, maximum and average number of objects in each warehouse of the system during the simulation time.
3. The load of all units in percent during the simulation time and the current load.

To create such a simulation model, you need to use the POST/api/v1/model service interface and put a JSON object describing model in the HTTP request body.

Figure 4 shows a small part of this object. You can see the first nodes of the model on it: two operations "Slab store" and "Batch generator" and the beginning of the description of the "In bake" agent.

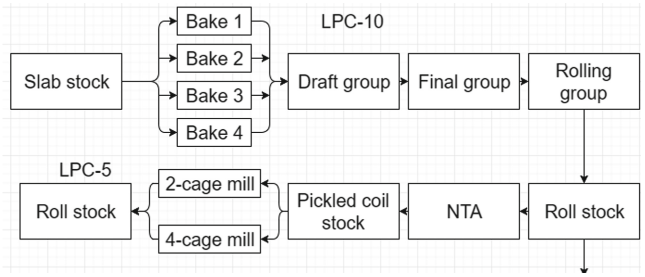
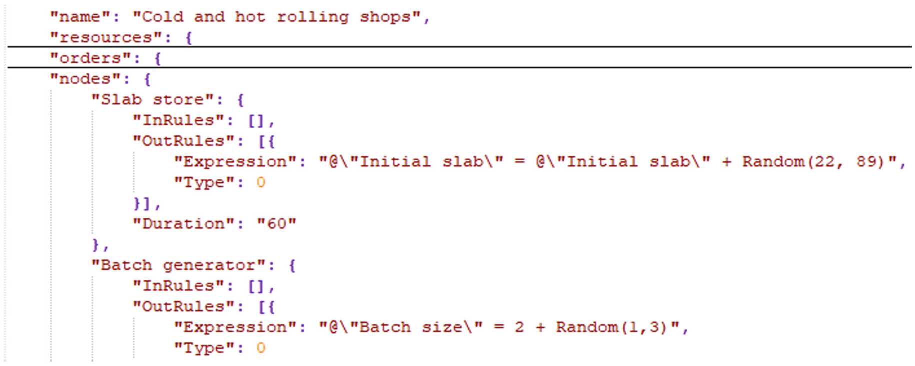


Fig. 3. Product flow chart between LPC-10 and LPC-5.



```
    "name": "Cold and hot rolling shops",
    "resources": {
    "orders": {
    "nodes": {
        "Slab store": {
            "InRules": [],
            "OutRules": [{
                "Expression": "@\"Initial slab\" = @\"Initial slab\" + Random(22, 89)",
                "Type": 0
            }],
            "Duration": "60"
        },
        "Batch generator": {
            "InRules": [],
            "OutRules": [{
                "Expression": "@\"Batch size\" = 2 + Random(1,3)",
                "Type": 0
```

Fig. 4. Part of the model description.

Nine experiments were conducted with the model. In each experiment, the minimum value of the slabs in the warehouse was changed. Table 3 presents a partial result of the experiments, including the minimum value of slabs in the warehouse, the final output of the model and the waiting time—the total idle time of the furnaces.

According to the results of the experiments, it was decided that the 6th experiment was the best. Starting from it, a continuous flow of slabs in workshops was obtained with the minimum quantity in the warehouse, the best effect was obtained.

Table 3. The results of experiments.

№	The min. number of slabs	Model output	Waiting time
1	310	776	234
2	350	628	133
3	390	538	69
4	410	543	65
5	430	529	48
6	477	503	11
7	490	517	13

8 Conclusion

The data obtained in the course of this work made it possible to analyze the current development of business process simulation systems (such as AnyLogic, ARIS, BPsim, G2) and highlight the requirements for a new system oriented to work on the Internet. A comparative analysis of the existing dynamic BP models was carried out and the model of the multi-agent resource conversion process was taken as a basis. A prototype web-service for BP simulation BPsim.Web was developed. The web service has been tested in solving the problem of analyzing the processes of two workshops.

Acknowledgment. The reported study was funded by RFBR according to the research project № 18-37-00183.

References

1. Devyatkov, V.V., Vlasov, S.A., Devyatkov, T.V.: Cloud technology in simulation studies, GPSS cloud project. In: Proceedings of the 7th IFAC Conference on Manufacturing Modeling, Management, and Control 2013, IFAC, vol. 7, pp. 637–641 (2013)
2. Solovyeva, I., Sokolov, B., Ivanov, D.: Analysis of position optimization method applicability in supply chain management problem. In: Proceedings of the Conference on Stability and Control Processes in Memory, pp. 498–500(2015)
3. Sokolov, B.V., Pavlov, A.N., Yusupov, R.M., Ohtilev, M.U., Potryasaev, S.A.: Theoretical and technological foundations of complex objects proactive monitoring management and control. In: Proceedings of the Symposium Automated Systems, pp. 103–110 (2015
4. Borodin, A., Kiselev, Y., Mirvoda, S., Porshnev, S.: On design of domain-specific query language for the metallurgical industry. In: Kozielski, S., Mrozek, D., Kasprowski, P., Małysiak-Mrozek, B., Kostrzewa, D. (eds.) BDAS 2015. CCIS, vol. 521, pp. 505–515. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18422-7_45
5. Aksyonov, K., Bykov, E., Aksyonova, O., Goncharov, N., Nevolina, A.: The architecture of the multi-agent resource conversion processes. In: Proceedings of the 11th European Modelling Symposium on Computer Simulation, pp. 61–64 (2017)

6. Aksyonov, K., Antonova, A., Goncharova, N.: Analysis of the electric arc furnace workshop logistic processes using multiagent simulation. In: Thampi, S.M., Krishnan, S., Corchado Rodriguez, J.M., Das, S., Wozniak, M., Al-Jumeily, D. (eds.) SIRS 2017. AISC, vol. 678, pp. 390–397. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-67934-1_35
7. Cao, L., Zeng, Y., Symeonidis, A.L., Gorodetsky, V., Müller, J., Yu, P. (eds.): Agents and Data Mining Interaction. Lecture Notes in Artificial Intelligence, vol. 8316. Springer, Cham (2014). https://doi.org/10.1007/978-3-642-55192-5
8. Wooldridge, M.: Intelligent agent: theory and practice. Knowl. Eng. Rev. **10**(2), 115–152 (1995)
9. Hammer, M., Champy, J.: Reengineering the Corporation: A Manifesto for Business Revolutions. Harper Business, New York (1993)
10. Hartmann, S., Drexl, A.: Project scheduling with multiple modes: a comparison of exact algorithms. Netw. Int. J. **32**(4), 283–297 (1998)
11. Osaba, E., Carballedo, R., Diaz, F.: Simulation tool based on a memetic algorithm to solve a real instance of a dynamic TSP. In: Proceedings of the IASTED International Conference Applied Simulation and Modelling, pp. 27–33 (2012)
12. Dreżewski, R.: A model of co-evolution in multi-agent system. In: Mařík, V., Pěchouček, M., Müller, J. (eds.) CEEMAS 2003. LNCS (LNAI), vol. 2691, pp. 314–323. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45023-8_30
13. Fillipovich, A.U.: Integration of Situational, Simulation and Expert Modeling Systems. Publisher “OOO Elix+”, Moscow (2003)
14. Moskaliov, I.M.: System for analysis and optimization of resource conversion processes. Thesis, Russia, Ekaterinburg (2006)
15. Klebanov, B., Antropov, T., Riabkina, E.: The principles of multi-agent models of development based on the needs of the agents. In: Proceedings of the 35th Chinese Control Conference, pp. 7551–7555 (2016)
16. Klebanov, B., Antropov, T., Riabkina, E.: Bases of imitation model of artificial society construction accounting of the agents’ needs recursion. In: Proceedings of the 16th International Multidisciplinary Scientific GeoConference, vol. 1, pp. 101–108 (2016)