ISSN 1847-3938

mipro 2017

organizer mipro

40th mipro 1978 2017

jubilee international convention

May 22-26, 2017, Opatija – Adriatic Coast, Croatia

Lampadem tradere

mipro - path to knowledge and innovation

mipro proceedings

Knowledge Elicitation in Multi-Agent System for Distributed Computing Management

Alexander Feoktistov* , Andrey Tchernykh**, Sergey Gorsky* and Roman Kostromin *
* Matrosov Institute for System Dynamics and Control Theory of SB RAS, Irkutsk, Russia
** Computer Science Department, CICESE Research Center, Ensenada, B.C., Mexico
agf65@yandex.ru, chernykh@cicese.mx, gorsky@icc.ru, romang70055@gmail.com

Abstract - The effective management of scalable applications for solving large problems in a heterogeneous distributed computing environment is the non-trivial problem. Scalable applications generate competitive job flows that have be executed with the help of shared resources of the environment. The promising approach to solve this problem is to use multi-agent technologies. To this end, we develop a multi-agent system for the management of scalable applications. In contrast to known multi-agent systems, our system is based on applying a special conceptual model of the environment. It includes several components of a comprehensive knowledge about both the environment and subject domains of solved problems. We propose a new approach to an elicitation of these knowledge components through an integrated use of the conceptual modelling of distributed computing, classification of jobs and resources, and parameters adjustment for agent algorithms. With this approach, specialists in various fields of distributed computing considered as users of the environment, can apply their own knowledge at different levels of the problem solving process. This flexibility and algorithm adaption are benefits of our approach. Extensive modeling and practical experiments, with variation of important parameters of applications execution, show the efficiency of our management under developed multi-agent system for scalable applications.

I. Introduction

High-performance computing (HPC) is an important component for effective solving large problems and mathematical modeling in various subject domains. HPC is based on the parallel execution of computational processes that are characterized different degrees of an interrelationship between them.

A rapid increase of the computational elements in systems for HPC leads to need for scalable applications. A computations scalability means that a problem solving time decreases with increasing a number of the nodes used by an application. Thus, the solved problem has to have an ability to be decomposed into sub problems, which can be solved as much independently as possible.

HPC supercomputers and clusters provide solutions within various distributed computing environments that include heterogeneous, autonomous and geographically distributed nodes that can have complex hybrid structures, special computational characteristics and administrative policies [1]. The nodes can have different owners with own criteria of the usage (efficiency, profit, load balancing, power consumption, etc.).

Grids [2] and Clouds [3] are typically form such an environment.

Resources management system (RMS) is used for computations management at the node level. The known traditional RMSs are Grid Engine [4], PBS Torque [5], HTCondor [6], and SLURM [7].

The specialized middleware is used for computations management of distributed nodes through an interaction with RMSs and their substitution. For example, the Globus Toolkit [8] supports various the standards and compatibility with various DRMSs, and GridWay [9].

Usually, the environment supports the collective usage of its resources by users for the large range of problem classes. Problem specification includes information about the required computational resources, executable programs, input and output data, quality criteria (time, cost, reliability, etc.), and other required issues. The specification describes a computational job for the traditional RMSs and meta-schedulers used in environment nodes.

In general, we assume that the job includes the set of interconnected programs. The job specifications are generated by applications in the form a workflow [10, 11, 12]. It can be represented by Directed Acyclic Graph (DAG) according to logical and information relations between the programs.

Workflow management systems (WMSs) largely used for solving scientific problems are known. There are the following systems: Condor DAGMan, Pegasus, Taverna, UNICORE, etc. [13]. Nevertheless, the workflow management problems in heterogeneous distributed computing environment is not completely solved [14].

One of problem is a concordance of all solving processes. WMS operates at the application level. Its objective is to obtain best resources for workflows.

Unlike WMSs, RMSs and meta-schedulers operate at the environment level. Node administrators can configure RMSs and meta-schedulers to determine common rules of a resources usage and policies for the allocation of quotas and rights for users and their jobs. However, the effective coordination of users and owners of the resources in processes of their allocation for solved problems, taking

into account the specificity of subject domains, often remains beyond their capabilities.

A promising approach for solving this non-trivial problem is applying a multi-agent technologies based on the self-organization of agent communities [15]. Within such an approach, a Multi-Agent System (MAS) implements the computations management. Agents can represent users and owners of the resources, and interact themselves with aim to meet their interests.

We know the range of MASs, successfully used for distributed computing management in practice [16,17]. Nevertheless, there are additional components of knowledge that do not used by these systems. They do not include the knowledge about the specific subject domain, job properties, agent algorithm parameters, etc.

To this end, we develop a multi-agent system for the management of scalable applications that allows specialists in various fields of distributed computing can apply their own knowledge at different levels of the problem solving process. The special attention is on a job classification system.

II. Multi-Agent System

The developed MAS has a hierarchical structure, which includes two or more functional levels, and operates on the base of self-organization. At each level, agents play a variety of roles, and perform different functions. The roles may be permanent or temporal. Their changes occur at discrete times when agents need to solve new problems. Each level is related with the conceptual model knowledge layers.

Agents are autonomous, and computing management is based on their interaction. They can be organized in virtual communities. In these communities, agents interact using a cooperation or competition. The formation of virtual communities allows the MAS to adapt the management process to the new challenges.

The MAS applies the algorithm for the static resource allocation based on economic mechanisms for regulation of resources demand and supply. It uses a model of the Vickrey auction with one-round bidding [18].

The users formulate problems and define criteria of their solving. User agents create and classify jobs based on the problem formulations. Then, these agents submit the jobs to agents responsible for resource allocation.

A virtual community of agents that most suitable for the problem solving is created. Agents use the classification system of resources and jobs for self-organization into the virtual community.

Each agent of the virtual community sets bids for all jobs that reflect the genuine value of these jobs and maximal utility of the bids for this agent. The steady state of the virtual community is achieved at the end of the auction. This state is similar to the equilibrium that has been defined by Nash [19] in the game theory. Rules of the Vickrey auction provides simplicity and satisfactory rate of the decision-making by agents.

According to results of bidding, the virtual community manage the job execution. In the process of computing, agents may reallocate its own computational load among other agents through interaction with their neighbors.

Agents can also provide dynamic decomposition of a problem into sub-problems, when additional free resources are available. The decomposition is carried out in order to accelerate the problem solving and make it scalable.

There are large variety of knowledge elicitation methods [20]. Conceptual and simulation modeling are major methods used by agents. Our implementation of these methods is based on conceptual programming [21, 22].

III. Conceptual Model

With high-level languages and methods, conceptual modeling becomes a powerful tool for complex systems design. A conceptual model has to allow the system designers and developers to represent and structure the results of their expert analysis of a subject domain.

In this regard, the conceptual model of a subject-oriented distributed computing environment includes the following components of comprehensive knowledge:

- Computational knowledge about software modules (programs) for solving problems in subject domains and working with the environment objects, including planning of their actions;
- Schematic knowledge about the modular structure of models and algorithms, problem formulations and workflows;
- Production knowledge to support decision-making for selecting optimal algorithms, depending on the workflow and environment states;
- Knowledge about the software and hardware infrastructure of the environment and administrative policies in its nodes, including an information about users, their problems and jobs.

We develop the specialized high-level toolkit named SIRIUS II for designing the conceptual models. This toolkit provides elicitation the subject-oriented knowledge of application developers. Detailed information about the conceptual model can be found in [23, 24].

IV. Job Classification System

We developed the job classification system to improve the resource allocation. This system allows to elicit knowledge of the administrators about a conformity of jobs and resources. Such a knowledge provides a possibility to determine a job class and select conformable resources.

Requirements to a computer system for a problem solving included in jobs represent explicit characteristics such as the problem solving time, RAM, disk memory, number of nodes, processors and cores, program libraries, compilers, their keys, etc.

Our job classification system allows the administrators to use these characteristics without changes, and develop new characteristics based on the value ranges for original characteristics. For example, they can create the new special characteristics with different ranges for the requested core number, problem solving time, and define job classes based on these characteristics.

Additionally, the administrators can develop implicit job characteristics using the conceptual model knowledge. These characteristics are based on the knowledge about methods, algorithms, programs execution conditions, computational history, etc.

A. Classification Model

A classification model has a set of explicit and implicit job characteristics $h_1, h_2, ..., h_k$. Each characteristic h_i for $i \in \overline{1,k}$ is defined by its rank $r_i \geq 1$, weight $w_i \geq 0$, and range R_i of values including the symbol θ of an uncertainty [25, 26]. Characteristics are partially descending ordered in accordance with their ranks.

We can define m job classes based on the set of characteristics. Each new class c_j is defined using the two disjoint subsets of characteristics: mandatory and optional subsets. The characteristic h_i included in one of these subsets for the class c_j has the specialized range $R_{ij}^{spec} \subseteq R_i \setminus \{\theta\}$ of values.

Let us denote by x and y the Boolean vectors of dimensions k and m, respectively.

Value of the element x_i is defined as follows:

$$x_i = \begin{cases} 0, if\ R_i^{job} \equiv \{\theta\}, \\ 1, if\ R_i^{job} \subseteq R_i \setminus \{\theta\}, \end{cases}$$

where R_i^{job} is the range of values, requested for the characteristic h_i. The value $x_i = 1$ ($x_i = 0$) shows that the characteristic h_i is used (not used) in the job specification. The vector x satisfies $\bigwedge_{i=1}^{k} \bar{x}_i = 0$.

Define the following characteristic function:

$$\chi_j(x) = \begin{cases} 0, if\ \exists i : (x_i = 1) \wedge (R_i^{job} \not\subseteq R_{ij}^{spec}), \\ 1 \text{ otherwise}, \end{cases}$$

where $i \in \overline{1,k}$, $j \in \overline{1,m}$. The value $\chi_j(x) = 1$ ($\chi_j(x) = 0$) means that the job is matched (mismatched) to the class c_j.

We implement a primary job classification applying the function $\chi_j(x)$ for m classes: $y_j = \chi_j(x)$, where $y_j = 1$ ($y_j = 0$), means that the job match (mismatch) to the class c_j, $j = \overline{1,m}$. It is obvious that the job can match to several classes.

To this end, we use the additional characteristic function $\varphi_{jl}(x, y)$ to determine when the job match ($\varphi_{jl}(x, y) = 1$) or mismatch ($\varphi_{jl}(x, y) = 0$) to the class c_j taking into account the four properties of job characteristics:

1. Probabilistic measure based on the number of optional characteristics matching to the class ($l = 1$);
2. Aggregated rank ($l = 2$);
3. Summary weight ($l = 3$);
4. Probabilistic measure based on the computational history of jobs ($l = 4$).

These functions allow implementing various variants of the additional job classification based on the primary job classification. The job classification intended for the primary filtration of the resources set. In general, it provides forming the residual set V of resources for the job execution.

B. Job Classification Algoritm

In the job classification system, the administrator creates the sets of characteristics and classes, and defines conformity of jobs and resources using own expert knowledge. The MAS uses the sorting method for the knowledge elicitation.

During the system operation, the administrator can modify and add resources and job classes including inheritance mechanisms usage.

When the job submitted in the MAS, agents apply the job classification algorithm described below.

Majority stages of the algorithm:

1. Initialization of the vector y: $y_j = 0$, $j = \overline{1,m}$.
2. Primary job classification: $y_j = \chi_j(x)$, $j = \overline{1,m}$.
3. If the vector y satisfies $\bigvee_{j=1}^{m} y_j = 0$, then the job classification is not possible. Go to the stage 6.
4. Otherwise, if the vector y satisfies $\left(\bigvee_{l=1}^{k-1} \bigvee_{q=l+1}^{k} (y_l \wedge y_q) \right) \vee \left(\bigwedge_{l=1}^{k} \bar{y}_l \right) = 0$, then the single element $y_j = 1$ exists and the job matches to the single class c_j. Go to the stage 6.
5. Otherwise, $y_j = \varphi_{jl}(x, y), \forall j : y_j = 1, l = \overline{1,4}$.
6. Completion of the algorithm.

After the job classification, the MAS specializes this job by adding the directive how to use the selected resources. The MAS creates the virtual community of agents corresponding to selected resources. Then, it submits the specialized job to these agents. They perform the final selection and allocation of resources.

V. AGENT ALGORITHM ADJUSTMENT

Algorithms for the resource allocation have the following control parameters:

- Limits of the node components load;
- Bonuses for limits satisfying;
- Penalties for limits exceeding;
- Priorities of job classes;
- Degree of desire to carry out the jobs of defined classes.

Administrators as experts set the initial values of these parameters in configuring the MAS. These parameters are automatically adjusted based on simulation modeling and meta-monitoring in a process of the MAS operation.

Figure 1 shows the block diagram of the parameter adjustment of agent algorithms.

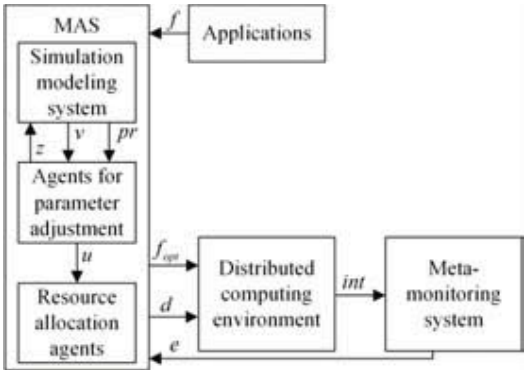

Figure 1. Block diagram of the parameter adjustment

Table I describes the parameters used on the block diagram.

TABLE I. PARAMETERS DESCRIPTION

Notation	Description
f	Job flow
f_{opt}	Optimized job flow
d	Job distribution
inf	Information about the environment state
e	Evaluation of the environment state
z	Vector of input variables of the simulation model
v	Vector of observed variables of the simulation model
pr	Prediction of the environment state changes
u	Vector of the control parameters

Agents for parameter adjustment use the simulation modeling to solve the following problem:

$$v_i(z) \to \min(\max), \quad (1)$$

$$v_i^{\min} \le v_i \le v_i^{\max}, \; i = \overline{1,n}. \quad (2)$$

We create the simulation model as the parameter sweep application that generates the large number of model copies [27]. Each independent copy runs with the single values variant of its input variables. A meta-monitoring system provides the actual evaluation of the environment state. After execution of all copies, the environment services provide the collection of simulation results represented by variants of the vector v. We apply multi-criteria selection of the results to form the vector u with evaluation of components of the vector v using the conditions represented in (1) and (2) [28, 29]. Administrators assign an extremum for each component of the vector v before the simulation modeling.

The parameter adjustment provides for agents the efficiency of their bids for jobs at the Vickrey auction.

We consider the parameter adjustment as the element of agent learning. A feedback for learning process is implemented using the bonuses and penalties. Agents correct own decision-making taking into account this feedback.

VI. EXPERIMENTAL ANALISYS

In this section, we provides results of extensive modelling and practical experiments with the developed MAS.

We obtained and analyzed the computational history about 80000 real jobs executed in one day on three computer clusters with different computational resources (Table II).

TABLE II. COMPUTER CLUSTERS

Cluster	Peak performance	Processor	Nodes/ processors/ cores numbers	RMS
Cluster 1	1.50 TFlops	Intel Xeon E5345	20/40/160	Cleo [30]
Cluster 2	0.77 TFlops	AMD Athlon II X4	16/16/64	HTCondor
Cluster 3	0.17 TFlops	Intel Xeon	16/32/32	SUPPZ [31]

Clusters 1 and Cluster 3 include dedicated nodes that are used within cluster only. Non-dedicated nodes of the Cluster 2 are used by owners and cluster users. In the Cleo, all cores of a node are allocated for a job request, regardless of the number of requested cores. Often, it leads to inefficient resource loading. We solve this problem using classification system.

In the classification system, we define six parental classes of jobs based on the analysis of computational history (Table III). These classes differ by the number of requested cores and execution time. Then, we created additional classes of jobs based on the parental classes taking into account the required RAM, disk space, system, applied software, etc. Each additional class is related to one of the clusters.

TABLE III. CLASSIFICATION RESULTS

Parental class	n_1	t	n_2	n_3
c_1	1	1 - 5	10221	4675
c_2	1	5 - 60	7927	5861
c_3	1	≥ 60	1211	972
c_4	≥ 2	1 - 5	33453	492
c_5	≥ 2	5 - 60	6125	50
c_6	≥ 2	≥ 60	3312	95

We generated the job flow in accordance to the obtained computational history. Each job is classified and submitted by the MAS to the matched cluster.

Table III presents the experimental results for 6 classes showing the following parameters:

- Number n_1 of requested cores,
- Requested execution time t in minutes,
- Number n_2 of jobs for the given class,
- Number n_3 of reallocation jobs.

When the classification system is used, about 12000 jobs have been relocated for better resource utilization in comparison to initial processing. The relocated jobs retained all characteristics of the classes defined for them. Figure 2 and Figure 3 illustrate a jobs distribution change for hours of the day without job classification and with it.

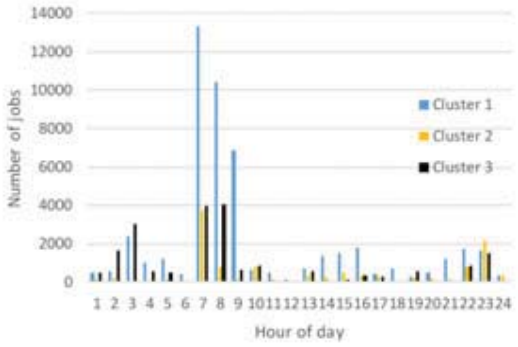

Figure 2. Jobs distribution with classification

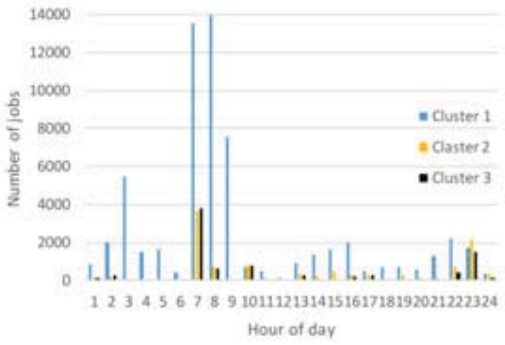

Figure 3. Jobs distribution without classification

The average processor load for three clusters during the job flow processing is improved by 18 %, 25 % and 30 % respectively (Figure 4).

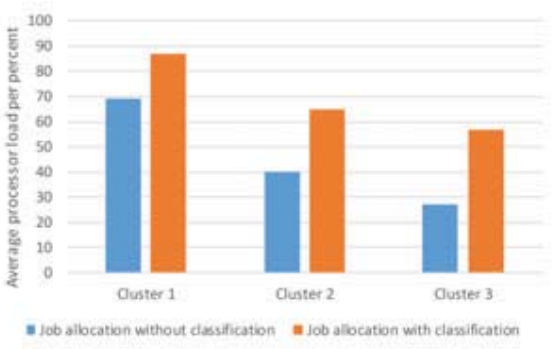

Figure 4. Jobs distribution with their classification

In the following experiment, the synthetic job flow is processed in the distributed computing environment including three clusters. The jobs represent a copy of the same workflow.

We compare the efficiency of processors load with management under the meta-scheduler of HTCondor version 6.6.8, GridWay version 5.12 and developed MAS, which as middleware interact with clusters RMSs.

We execute an application for warehouse logistics problem solving. Figure 5 illustrates the advantages of the MAS in comparison with target systems for this example. These advantages are achieved due to the control parameters adjustment represented in the Section V.

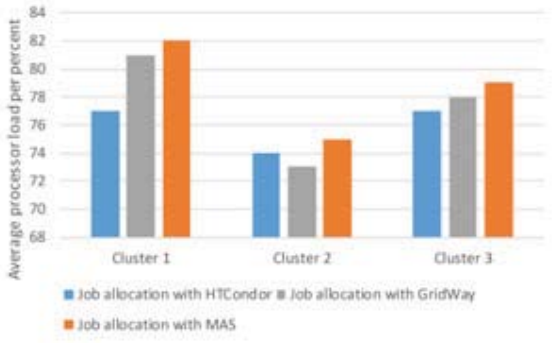

Figure 5. Cluster load with three management systems

Experiments is carried out in the distributed computing environment based on resources of the Irkutsk state university and Center of collective usage "Irkutsk Supercomputer Center of SB RAS" [32].

VII. CONLUSION

In this paper, we address a problem of the multi-agent management in distributed computing. We showed that its efficiency significantly depends on ability to use the subject-oriented knowledge.

We propose the approach to elicit, represent and use the comprehensive knowledge about both the distributed computing environment and subject domains of solved problems. Specialists in various fields of distributed

computing can generate this knowledge at different levels of the problem solving process based on own skills and experience. We also developed the MAS for the management of scalable applications that generate job flows. The MAS operates based on the self-organization.

Comprehensive modelling and practical experiments, with variation of important parameters, show the efficiency of our management under the developed MAS due to elicitation and using the conceptual knowledge.

As future work, we will continue the MAS evolution, expanding the subject-oriented knowledge usage.

ACKNOWLEDGMENT

The study is partially supported by Russian Foundation of Basic Research, projects no. 15-29-07955 and no. 16-07-00931, and Program 1.33P of fundamental research of Presidium RAS, project "Development of new approaches to creation and study of complex models of information-computational and dynamic systems with applications".

REFERENCES

[1] L. Zeng, L. Xu, Z. Shi, M. Wang, W. Wu, "Distributed Computing Environment: Approaches and Applications," IEEE International Conference on Systems, Man and Cybernetics, IEEE Publisher, pp. 3240-3244, October 2007.

[2] I. Foster, C. Kesselman, "Computational Grids," in The Grid: Blueprint for a New Computing Infrastructure, I. Foster, C. Kesselman, Eds. San Francisco: Morgan Kaufmann, pp. 2-48, 1999.

[3] R. Buyya, C. Vecchiola, S. T. Selvi, "Mastering Cloud Computing," Burlington, Massachusetts: Morgan Kaufmann, 2013.

[4] "Oracle Grid Engine," http://www.oracle.com/technetwork/oem/grid-engine-166852.html [online, accessed: Jan 31 2017].

[5] "Torque Resource manager," http://www.adaptivecomputing.com/products/open-source/torque/ [online, accessed Jan 31 2017].

[6] "HTCondor," http://research.cs.wisc.edu/htcondor/ [online, accessed: Jan 31 2017].

[7] "Slurm Workload Manager," http://slurm.net/ [online, accessed: Jan 31 2017].

[8] "Globus Toolkit Homepage," http://www.globus.org/toolkit/ [online, accessed Jan 31 2017].

[9] "GridWay Metascheduler," http://www.gridway.org [online, accessed Jan 31 2017].

[10] A. Rodriguez, A. Tchernykh, K. Ecker, "Algorithms for Dynamic Scheduling of Unit Execution Time Tasks," European Journal Operation Research, vol. 146, no. 2, pp. 403-416, April 2003.

[11] D. Kliazovich, J. E. Pecero, A. Tchernykh, P. Bouvry, S. U. Khan, A. Y. Zomaya, "CA-DAG: Modeling Communication-Aware Applications for Scheduling in Cloud Computing," Journal of Grid Computing, vol. 14, no. 1, pp. 22-39, March 2016.

[12] A. Hirales-Carbajal, A. Tchernykh, T. Roblitz, R. Yahyapour, "A Grid simulation framework to study advance scheduling strategies for complex workflow applications," IEEE International Symposium on Parallel and Distributed Processing, Workshops and Phd Forum (IPDPSW), IEEE Publisher, pp. 1-8, May 2010.

[13] J. Yu, R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing," Journal of Grid Computing, vol. 3, no. 3, pp. 171-200, September 2005.

[14] D. Talia, "Workflow Systems for Science: Concepts and Tools," ISRN Software Engineering, vol. 2013, 2013, http://dx.doi.org/10.1155/2013/404525 [online, accessed: January 31 2017].

[15] G. Di Marzo Serugendo, M.-P. Gleizes, A. Karageorgos, "Self-organization in Multi-agent Systems," The Knowledge Engineering Review, vol. 20, no. 2, pp. 165-189, June 2005.

[16] D. Talia, "Cloud Computing and Software Agents: Towards Cloud Intelligent Services," 12th Workshop on Objects and Agent, CEUR Workshop Proceedings, vol. 741, pp. 2-6, June 2011.

[17] P. Leitao, U. Inden, C.-P. Ruckemann, "Parallelising Multi-agent Systems for High Performance Computing," 3rd International Conference on Advanced Communications and Computation, Red Hook, NY: IARIA, pp. 1-6, June 2014

[18] W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders," Journal of Finance, vol. 16, no. 1, pp. 8-37, March 1961.

[19] J. Nash, "Equilibrium points in n-person games," Proceedings of the National Academy of Sciences of the United States of America, vol. 36, no. 1, pp. 48-49, 1950.

[20] N. J. Cooke "Varieties of knowledge elicitation techniques," International Journal of Human-Computer Studies, vol. 41, no. 6, pp. 801-849, December 1994.

[21] J. Sowa, "Conceptual Structures – Information Processing in Mind and Machine," in The Systems Programming Series, Addison-Wesley, 1984.

[22] E. Tyugu, "Knowledge-Based Programming (Turing Institute Press Knowledge Engineering Tutorial Series)," Boston: Addison-Wesley, 1988.

[23] A. G. Feoktistov, I. A. Sidorov, "Logical-Probabilistic Analysis of Distributed Computing Reliability," 39th International Convention on information and communication technology, electronics and microelectronics, Riejka: MIPRO, pp. 247-252, May 2016.

[24] I. Bychkov, G. Oparin, A. Tchernykh, A. Feoktistov, V. Bogdanova, S. Gorsky, "Conceptual Model of Problem-Oriented Heterogeneous Distributed Computing Environment with Multi-Agent Management," Procedia Computer Science, vol. 103, pp. 162-167, January 2017.

[25] A. Tchernykh, U. Schwiegelsohn, V. Alexandrov, E. Talbi, "Towards Understanding Uncertainty in Cloud Computing Resource Provisioning," Procedia Computer Science, vol. 51, pp. 1772–1781, May 2015.

[26] A. Tchernykh, U. Schwiegelsohn, E. Talbi, M. Babenko, "Towards Understanding Uncertainty in Cloud Computing with risks of Confidentiality, Integrity, and Availability," Journal of Computational Science, http://www.sciencedirect.com/science/article/pii/S1877750316303878 [online, accessed: January 31 2017].

[27] H. Casanova, F. Berman, G. Obertelli, R. Wolski, "The apples parameter sweep template: User-level middleware for the grid," ACM/IEEE conference on Supercomputing, Washington: IEEE Press, pp. 111-126, November 2000.

[28] I. V. Bychkov, G. A. Oparin, A. G. Feoktistov, V. G. Bogdanova, A. A. Pashinin, "Service-oriented multiagent control of distributed computations," Automation and Remote Control, vol. 76, no. 11, pp. 2000-2010, November 2015.

[29] I. V. Bychkov, G. A. Oparin, A. G. Feoktistov, I. A. Sidorov, V. G. Bogdanova, S. A. Gorsky, "Multiagent Control of Computational Systems on the Basis of Meta-monitoring and Imitational Simulation," Optoelectronics, Instrumentation and Data Processing, vol. 52, no. 2, pp. 107-112, June 2016.

[30] "Cleo Cluster Batch System," https://sourceforge.net/projects/cleo-bs.html [online, accessed: January 31 2017].

[31] SUPPZ http://suppz.jscc.ru/ [online, accessed: January 31 2017].

"Irkutsk Supercomputer Centre of SB RAS," http://hpc.icc.ru [online, accessed: January 31 2017].