Automated tools for the development of microservice compositions for hybrid scientific computations

G A Oparin, V G Bogdanova and A A Pashinin

Matrosov Institute for System Dynamics and Control Theory of SB RAS, Lermontov St. 134, Irkutsk, Russia, 664033

bvg@icc.ru

Abstract. In recent years, a significant amount of research is focused on the development of tools for creating composite web-services for solving both business and scientific complex problems. This study discusses tools for building compositions or ensembles of microservices (depending on the method of integration) developed based on the HPCSOMAS framework. These tools are oriented on the application in a package of applied microservices for solving computationally complex problems of structural analysis and parametric synthesis of controlled dynamic systems in a heterogeneous high-performance computing environment. In particular, binary dynamic systems are studied using the Boolean constraint method for both their qualitative analysis and synthesis of laws to control these systems. Creating and executing composite services is carried out on a semantic peer-to-peer network of agents. The HPCSOMAS framework supports two modes of these processes, both the static creation and application of a composite service based on the procedural formulation of the problem and dynamic, based on the declarative formulation. In the first case, agents deployed on the network perform hierarchical control over the execution of the composition of microservices, in the second case, decentralized asynchronous management of the ensemble of microservices. Both operating modes are automated, and the validity of the resulting composite service is checked based on a logical approach. The tools are aimed both at a professional programmer and the end-user, a specialist in the subject domain. The HPCSOMAS framework supports the execution of composite microservices in a hybrid computing infrastructure, which includes both cloud and on-premises resources.

1. Introduction

The complexity of the modern heterogeneous distributed computing environment (HDCE) prevents the widespread use of its capabilities by specialists in the subject domain. Therefore, the demanded trend is the development of specialized approaches that free the end-user from the need to study the technical details for the creation of distributed service-oriented applications. As a result, in recent years, a considerable number of research efforts focus on the development of frameworks for creating composite web services for solving both business and scientific complex problems [1]. One of the advantages of service-oriented computing is the ability to combine several services to get more meaningful and complex ones [2, 3]. A composite web service can be created either proactively or reactively [4]. In the first case, components of the composite web service are assembled in advance, before runtime. In the second case, a composite web service is created dynamically at the request of the user. This request is executed in the ever-changing HDCE, so users need more adaptive and personalized services [5]. The most general standards and research approaches developed over the last years for web service composition are described in [6]. New challenges concerning this problem are mentioned in this publication also, in particular, challenges of automated RESTful services composition and its orientation to the subject domain.

Our research is focused on the development of service-oriented applications based on HPCATAMP technology. These applications are designed for solving computationally complex problems of the subject domain of structural analysis and parametric synthesis of controlled dynamic systems in HDCE [7]. In particular, binary dynamic systems (BDS) are studied using the Boolean constraint method [8] for both their qualitative analysis and synthesis of laws to control these systems. Currently, there is an intensive development and complication of controlled dynamic systems. Despite a large number of existing methods [10], the problem of synthesis of control laws for various classes of these systems remains relevant.

The above HPCATAMP technology is implemented based on the HPCSOMAS platform [10], the basic version of which is periodically modified based on experience in solving problems from the above domains, for example, [11-13]. In this study, we describe the multi-agent tools of the cloud version of this HPCSOMAS-MSC platform (version HPCSOMAS 3.2) for creating composite services in the form of an ensemble of microservices implemented based on the RESTful architectural style. Compositions and ensembles of microservices are used when performing computations in the package of applied microservices (AMP [7]). As an illustrative example, we describe the application of composite services of the AMP for solving problems of qualitative analysis of BDS.

2. Related work

In review papers [3, 15-20], various issues of the development of composite services are considered. As noted in [3], most approaches are based on the composition of the workflow (WF) or methods of artificial intelligence (AI), in particular, AI-planning. The paper [3] mainly addresses the issues of comparing dynamic composition based on these two approaches. In general, the WF approach is used in the situation when a process model is already defined in the user request, so the search for atomic services may be automated. AI-planning methods are used when the user request does not have a process model but has many constraints and user preferences. In this case, a process model is automatically generated. In [15], a consolidated structure of the analysis of models, languages, methods, platforms, and tools for the composition of services is presented. Several representative systems (in particular, [21-23]) that automate the composition of services were analyzed in [15] based on criteria of application, notation (visual or textual), paradigm (Flow-based, Rule-based, Query-based), composition constructs, and the target user. Most systems are based on the Flow paradigm, focused on business processes. The target users of the analyzed systems are mainly professional programmers, and only three of the twelve systems are aimed at the end user-programmer, an expert in the subject area. In [15], there is a need for the development of compositional tools that can be mastered not only by professionals. The necessity of extending the composition of services and developing composition methods to ensure the integration of cloud resources with heterogeneous high-performance environments is substantiated. Despite the variety of tools based on the WF approach presented in later publications [24–26], some questions remain relevant, in particular, the target user qualification, decentralized control, adaptive execution models, high-level composition tools, fault-tolerance. An alternative to the WF approach is the approach based on AI-planning [3]. In [16], the advantages and disadvantages of AI methods are considered. In particular, it is noted that AI-planning is suitable for dynamically creating web services with incomplete information, but the main problem in planning is scalability. Also, this planning is unsuitable for web services composition based on choreography, which implies decentralized control, parallel workflows, and unexpected circumstances. Issues related to the compatibility and the provision of portability for the efficient composition of web services in heterogeneous and decentralized environments also remain open. In [17, 18], AI-planning is performed based on decentralized multi-agent control. However, after the construction of the plan, its execution is centralized. In [19], the development of a service composition system based on the declarative description and usage of the peer-to-peer (P2P) paradigm that provides the execution of the resulting composite services based on decentralized control is presented. In [20], a review of existing methods, approaches, and standards for web services discovery and composition development were performed. As noted in this publication, the verification of

composition validity is unused in most of the considered approaches for composition development, so the composition specifications in the form of mathematical formalisms are desirable for such verification.
Given the above advantages and disadvantages of existing approaches, we use a hybrid approach, providing the developed tools with the ability to carry out the microservices composition in both static and dynamic ways. This approach is based on the multiagent-based P2P network and microservice-oriented technology. Under the composite service, we mean the composition of microservices in static mode and the ensemble of microservices in dynamic. Composite services based on the proposed approach are designed to solve the above scientific problems and are focused on the use in a hybrid computing infrastructure, which includes both on-premises and cloud resources.

3. Creating composite service based on HPCSOMAC-MSC

Depending on the particular computing problems of the subject domain, as well as the requirements and preferences of the user, the HPCSOMAS-MSC platform provides two modes for creating a composite service, static and dynamic (figure 1). In the first case, based on the procedural formulation of the problem, a composite service is fully prepared, and only then it is launched. In the second case, for the non-procedural problem statement (NPS), the ensemble of microservices is composed dynamically in the process of computing. The dynamic composition of web services is a promising approach [8]. This approach is in demand when changes to the runtime requirements are frequent, and when a complete set of services cannot be specified during development [6]. The dynamic composition provides automating problems associated with the assembly of microservices. However, it limits the freedom of users in the process of this assembly [6]. The static one is suitable in the case of the functional requirements for the composite service remain unchanged for a long time [6]. When solving problems in the subject domain mentioned in the introduction, both of these cases may occur. Therefore, the HPCSPMAS-MSC framework supports both the two modes, static and dynamic creation and execution of composite services. The static mode is provided by the AMP manager-agent (AMPMA) and computational microservices agents (CMA). For dynamic one, the AMPMA initializes distributed computational agents (DCA). All agents are created during the AMP development based on HPCSOMAS-MSC programming or automating tools.

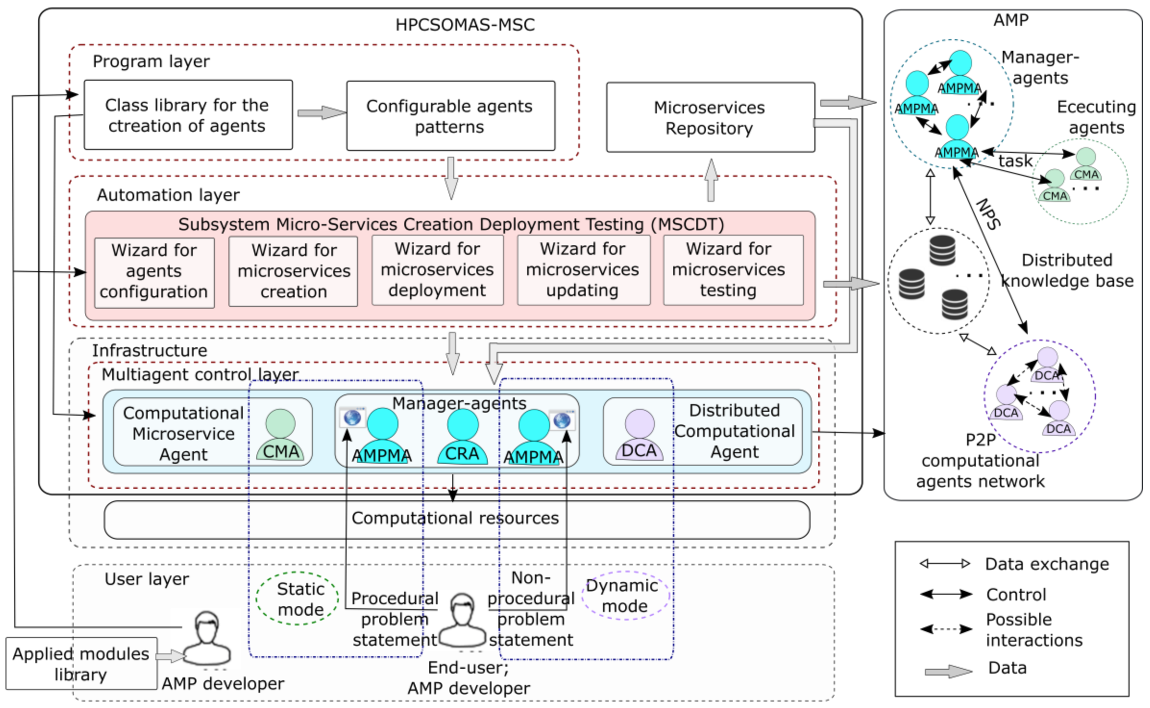

HPCSOMAS-MSC
Program layer
Class library for the ctreation of agents
Configurable agents patterns
Microservices Repository
Automation layer
Subsystem Micro-Services Creation Deployment Testing (MSCDT)
Wizard for agents configuration
Wizard for microservices creation
Wizard for microservices deployment
Wizard for microservices updating
Wizard for microservices testing
Infrastructure
Multiagent control layer
Computational Microservice Agent
CMA
Manager-agents
AMPMA
CRA
AMPMA
DCA
Distributed Computational Agent
Computational resources
User layer
Static mode
Procedural problem statement
Non-procedural problem statement
Dynamic mode
Applied modules library
AMP developer
End-user; AMP developer
AMP
Manager-agents
AMPMA
AMPMA
AMPMA
Ececuting agents
task
CMA
CMA
NPS
Distributed knowledge base
DCA
DCA
DCA
P2P computational agents network
Data exchange
Control
Possible interactions
Data

Figure 1. The HPCSOMAS-MSC architecture and AMP components.

3.1. Static composition of the microservices

The static composition is performed in several stages in the procedural formulation of the problem (figure 2):

- The user sends a request to the AMPMA, in which the sequence of microservices for solving the problem (microservice composition scheme) is given. The description of AMP microservices functionality is provided by the AMPMA. Additionally, the user specifies non-functional requirements.
- Based on the semantic description of microservices stored in the knowledge base (KB) in the form of fragments of the relations $In \subset P \times M$ and $Out \subset M \times P$ of microservices (*M*) with parameters (*P*), the AMPMA performs information planning, which is a verification of the admissibility of the microservice composition scheme. In the scheme $\{M_1(;A^1), M_2(A^2;B^2), ..., M_{n-1}(A^{n-1};B^{n-1}), M_n(A^n;)\}$, where n is the number of microservices, the AMPMA adds two microservices M_1 and M_n for, correspondently, sending input data (the set A^i) and receiving output data (the set B^i).
- The search for the microservices required by the scheme is performed on the AMPMAs installed on the HDCE nodes.

The validity of the microservices composition is checked according to the following condition. A composition scheme is admissible if, for any input parameter of any microservice in this scheme, there is exactly one previous microservice with the same output parameter. The following Boolean equation corresponds to this condition of the composition admissibility [27]:

$$\bigvee_{t=2}^{n} \bigvee_{p=1}^{n} (s_{tp} \wedge y) = 0\,, \tag{1}$$

where

$$y = \begin{cases} \bigvee_{q \in A_p} \left(\bigvee_{r=1}^{l-1} \bigvee_{g=r+1}^{l} (z_r \wedge z_g) \vee \bigwedge_{r=1}^{l} \overline{z_r} \right), \text{if } (A_p \neq 0) \wedge ((\forall q \in A_p)(B_q^{'} \neq 0)) \\ 1, \text{if } (A_p \neq 0) \wedge ((\exists q \in A_p)(B_q^{'} = 0)) \\ 0, \text{if } A_p = 0. \end{cases}$$

In equation (1), the matrix *S* is a matrix of the dimension $n \times n$ of Boolean variables s_{ij}. The *S* describes the execution scheme of the composite microservice. The element $s_{ij} = 1$ if the microservice $M_j \in M$ ($j = \overline{1,n}$) is in the *i*-th string of the matrix *S*. Matrices *A* and *B* are Boolean matrices of the dimension $n \times m$. Elements of these matrices are formed as follows: $a_{ij} = 1$ ($b_{ij} = 1$) if the parameter P_j is the input (output) one for the microservice M_j. Let A_i, B_i and $A_i^{'}, B_i^{'}$ ($i = \overline{1,n}$) are correspondently the strings and columns of matrices *A* and *B*. The *m* is the number of parameters of the execution scheme *S*. The values of strings and columns are zero if unit values are absent in these matrices elements. The *z* is the array of the length *l* consisting of elements of the matrix *S*. The array z is formed as follows:

$$z = \{s_{ij} : i = \overline{1, t-1}, j \in B_q^{'}\}\,.$$

Microservices have the following types:

- Atomic, performing one specific computational function,
- Simple, consisting of an atomic microservices sequence,
- Flow Pattern, used by the computational microservices agent (CMA) to control the flow of tasks,
- Composite, consisting of a composition of the above microservices and embedded composite microservices.

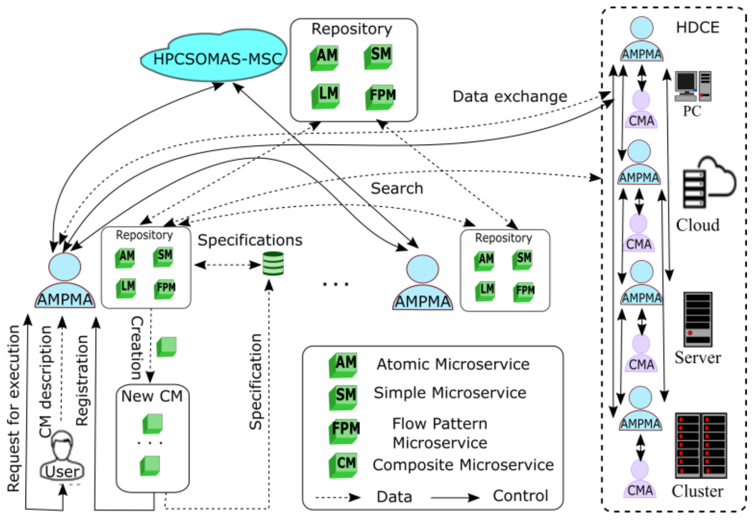


Figure 2. The static composition.

The microservice composition scheme describes a plan for conducting a computational experiment to solve a specific problem of a subject domain. This scheme is built based on a computational model of a subject domain and organizes the logic of execution of the microservice composition using task flow control templates. As templates, ones similar to basic control flow patterns and advanced branching and synchronization patterns [28] are used. A hierarchical control system is used to interpret the composite service. CMAs that run Flow Pattern microservices distribute tasks on the computational layers during execution. For example, for the parallel split template, the parallel subtasks are executed at this layer. Double numbering (<layer number>.<New layer number>) allows the dynamic creation of a new layer between two existing layers. Control by layer subtasks execution is carried out by the agent creating this layer. After the layer subtasks are completed, the agent returns control higher in the hierarchy.

3.2. Dynamic creating the microservices ensemble

Based on the dynamic approach, the microservice ensemble is assembled by a group of Distributed Computational Agent (DCA) deployed in a P2P semantic network. When registering a microservice on DCA, the description of the microservice interface is recorded in the local agent KB. All DCAs have the same behavioral model based on the agent’s knowledge and reactions to external and internal events. The dynamic assembly of microservices is performed as follows (figure 3):

- The user performs an NPS in the AMPMA interface.
- The AMPMA checks the admissibility and redundancy of the NPS. At the first stage, by a distributed logical inference method developed by the authors based on the “direct wave” method, the possibility of solving the problem on a distributed on HDCE nodes KB is checked. In the case of solvability, an active DCA group is formed. Agents of this group control the launch of microservices from the ensemble of microservices corresponding to this group. The set of ensemble microservices are redundant if some input parameters of the NPS do not affect the achievement of the goal. In the second stage, such parameters are excluded from the NPS using the distributed backward-wave method applied to the set of output parameters. The implementation of the distributed logical inference method based on the discrete-event finite-state model FSMwVW (Finite State Machine with Variables and Works)

of an agent behavior is presented in [29]. The modified model FFSMwVW is described in [30].

- A cooperative solution of the problem is performed by agents based on decentralized asynchronous control (on input data readiness) by direct P2P interactions defined by fragments of the *In* and *Out* relations specified in the local DCA KB.

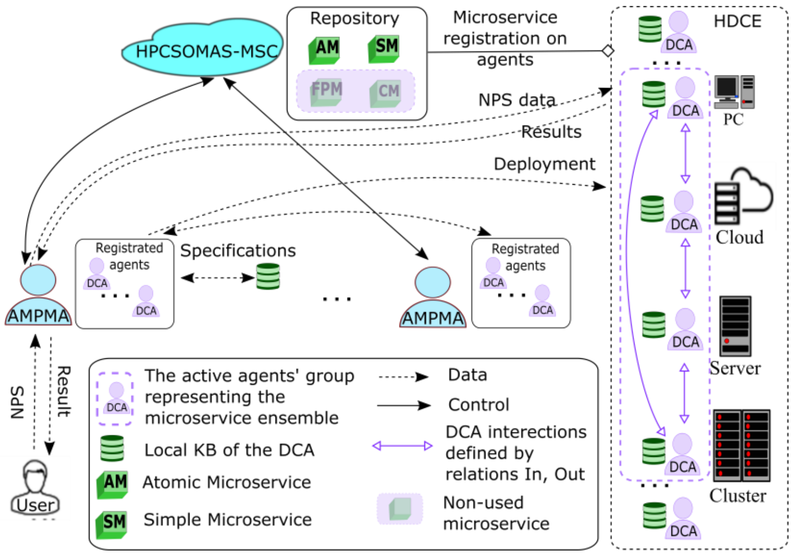


Figure 3. The dynamic composition.

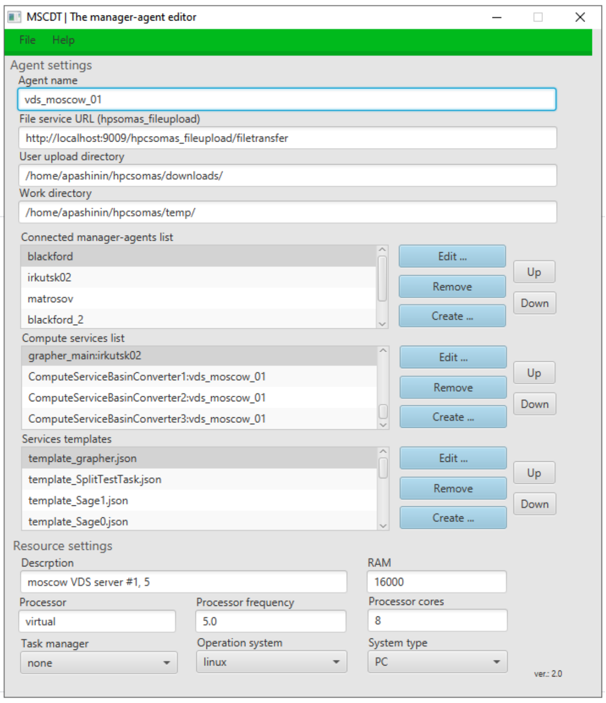
4. Illustrative example

In an illustrative example, composite services are used in the developed by authors AMP BCM-QABDS for qualitative analysis of BDS based on the Boolean constraint method (BCM, [8]). The computations are performed in a hybrid infrastructure, resources of which are given in table 1. The resource integration in this infrastructure is carried out by the installation of AMPMAs on this resource. The AMPMA must be configured after the installation. This configuration is set in the MSCDT [31] agent editor (figure 3). In figure 3, the configuring the AMPMA on the VDS resource is shown. The AMPMA has neighboring (connected agents) AMPMAs, the set of CMAs that can be seen in the scrolling list (*Compute services list* in figure 4).

Table 1. HDCE infrastructure.

Resource	Resource type	Installed software	
		Agents	Additional software
Nodes of cluster «Akademik V.M. Matrosov» [32]	On-premises	AMPMA	Tomcat Server
VDS [33]	Private Cloud	AMPMA	SageMath VirtualBox Docker Server Docker Container
VDS [33]	Public Cloud	CRA	Kernel Virtual Machine Tomcat Server
PC	On-premises	Dew-AMPMA	Tomcat Server

AMPMA with the microservice for building the Boolean model, verifying its feasibility (sequential SAT and 2QBF solvers), and post-processing the results can be installed on on-premises computers. Parallel solvers are installed on on-premises high-performance clusters. Additional resources are allocated using Cloud Resources Agent (CRA) from the public cloud (VDS for the example).



MSCDT | The manager-agent editor

Agent settings

Agent name: vds_moscow_01

File service URL (hpsomas_fileupload): http://localhost:9009/hpcsomas_fileupload/filetransfer

User upload directory: /home/apashinin/hpcsomas/downloads/

Work directory: /home/apashinin/hpcsomas/temp/

Connected manager-agents list:
- blackford
- irkutsk02
- matrosov
- blackford_2

Compute services list:
- grapher_main:irkutsk02
- ComputeServiceBasinConverter1:vds_moscow_01
- ComputeServiceBasinConverter2:vds_moscow_01
- ComputeServiceBasinConverter3:vds_moscow_01

Services templates:
- template_grapher.json
- template_SplitTestTask.json
- template_Sage1.json
- template_Sage0.json

Resource settings

Descrption: moscow VDS server #1, 5

RAM: 16000

Processor: virtual

Processor frequency: 5.0

Processor cores: 8

Task manager: none

Operation system: linux

System type: PC

ver.: 2.0

Figure 4. The interface for agents configuring.

As an example of applying a composite service created based on a dynamic approach, we consider the problem of searching for attractors and their basins in an autonomous asynchronous BDS.

In [1], we consider a synchronous autonomous BDS, the vector-matrix equation of which has the form

$$x^t = F(x^{t-1}), \qquad (2)$$

where x is the state vector of the dimension n ($x \in B^n, B = \{0,1\}$), $t \in T = \{1,2,\ldots,k\}$ is the discrete time, and the $F(x)$ is the vector-function of logic algebra called the transition function ($F: B^n \to B^n$). Let us define the trajectory $x(t, x^0)$ of (1) for each initial state $x^0 \in B^n$ as the finite sequence of states $x^0, x^1, \ldots, x^k$ from the set B^n. The state x^i is a successor of the state $x^{i\text{-}1}$, and $x^{i\text{-}1}$ is a predecessor of the state x^i. In an autonomous BDS, each state has only one successor, and the number of predecessors of this state can vary from zero to $2^{n\text{-}1}$. A cycle of the length k is a closed trajectory ($x^0 = x^k$) in which all other states are pairwise distinct. An equilibrium state is the cycle of the length k=1. A cycle is called an isolated if predecessors are absent for it.

The non-isolated cycle is called an attractor. Let X^* be the attractor. The region of attraction (basin) of an attractor of the radius k is the set of all states, from which the set X^* is reachable in $i \le k$ time steps. Equation (2) is equivalent to a single Boolean equation of the form

$$\Phi_k(x^0, x^1, \ldots, x^k) = \vee_{t=1}^{k} \vee_{i=1}^{n} (x_i^t \oplus F_i(x^{t-1})) = 0, \qquad (3)$$

where x_i^t and F_i are i-th components of vectors x^t and F; $\oplus$ denotes modulo-2 addition. For one-step transition ($k = 1$), equation (3) takes the form

$$\Phi_1(x^0, x^1) = \vee_{i=1}^{n} (x_i^1 \oplus F_i(x^0)) = 0. \qquad (4)$$

Taking into account (4), equation (3) can be rewritten as:

$$\Phi_k(x^0, x^1, \ldots, x^k) = \vee_{t=1}^{k} \Phi_1(x^{t-1}, x^t) = 0.$$

Problem statement. It is required to find attractors and their basins for the given BDS.
The Boolean models of BDS dynamical properties required for solving this problem are given in table 2. These models were constructed based on the BCM. In figure 4, the fragment of the computation model for solving the above problem is given.

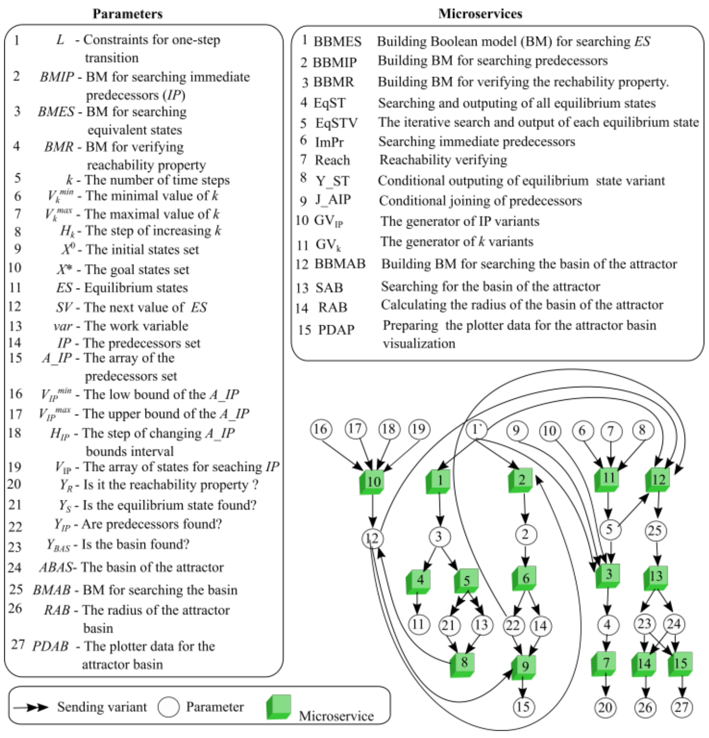
Table 2. Required Boolean models.

Problem	Boolean model	Result of the satisfiability checking	Interpretation of the result
Searching for equilibrium states	$\Phi_1(x^0,x^1)\big|_{x^1=x^0}=0$	SAT (UNSAT)	Equilibrium states are found (is not found)
Searching all immediate predecessors x^0 of the state $s \in B^n$	$\Phi_1(x^0,x^1)\big|_{x^1=s}=0$	SAT (UNSAT)	The *s* is an attractor (isolated equilibrium state)
Searching the attraction region (basin) of the attractor X^* of the radius *k*	$\Phi_k(x^0,x^1,...,x^k)\vee G^*(x^k)=0$ concerning to variables $x^{k-1},x^{k-2},...,x^0$	SAT (UNSAT)	The basin is found (is not found)

In table 2, the equation $G^*(x^k)=0$ determines a set of goal states [8].
For the dynamic approach, the following NPS is formulated using the web-interface of the AMPMA:

$$T_1 = (\,A_1^0 = \{L,k\}; B_1^0 = \{A\_IP, ABAS, Y\_BAS\})\,.$$

The meanings of input parameters A_1^0 and output parameters B_1^0 are given in figure 5. As a result of the forming ensemble stage, the microservices ensemble is assembled from the following microservices: {1, 2, 5, 6, 8, 9, 12, 13}.



Parameters

1	L	- Constraints for one-step transition
2	*BMIP*	- BM for searching immediate predecessors (*IP*)
3	*BMES*	- BM for searching equivalent states
4	*BMR*	- BM for verifying reachability property
5	k	- The number of time steps
6	V_k^{min}	- The minimal value of *k*
7	V_k^{max}	- The maximal value of *k*
8	H_k	- The step of increasing *k*
9	X^0	- The initial states set
10	X^*	- The goal states set
11	*ES*	- Equilibrium states
12	*SV*	- The next value of *ES*
13	*var*	- The work variable
14	*IP*	- The predecessors set
15	*A_IP*	- The array of the predecessors set
16	V_{IP}^{min}	- The low bound of the *A_IP*
17	V_{IP}^{max}	- The upper bound of the *A_IP*
18	H_{IP}	- The step of changing *A_IP* bounds interval
19	V_{IP}	- The array of states for seaching *IP*
20	Y_R	- Is it the reachability property ?
21	Y_S	- Is the equilibrium state found?
22	Y_{IP}	- Are predecessors found?
23	Y_{BAS}	- Is the basin found?
24	*ABAS*	- The basin of the attractor
25	*BMAB*	- BM for searching the basin
26	*RAB*	- The radius of the attractor basin
27	*PDAB*	- The plotter data for the attractor basin

Microservices

1	BBMES	Building Boolean model (BM) for searching *ES*
2	BBMIP	Building BM for searching predecessors
3	BBMR	Building BM for verifying the rechability property.
4	EqST	Searching and outputing of all equilibrium states
5	EqSTV	The iterative search and output of each equilibrium state
6	ImPr	Searching immediate predecessors
7	Reach	Reachability verifying
8	Y_ST	Conditional outputing of equilibrium state variant
9	J_AIP	Conditional joining of predecessors
10	GV_{IP}	The generator of IP variants
11	GV_k	The generator of *k* variants
12	BBMAB	Building BM for searching the basin of the attractor
13	SAB	Searching for the basin of the attractor
14	RAB	Calculating the radius of the basin of the attractor
15	PDAP	Preparing the plotter data for the attractor basin visualization

Figure 4. The fragment of the computational model.

We use the NPS T_1 for searching basins of attractors for the given BDS of the dimension n=11 using the example of gene regulator network from [34], dynamic of which is described by the following equations:

$$
\begin{aligned}
&x_1^t = x_1^{t-1}, x_2^t = x_2^{t-1} \cdot \overline{x_8^{t-1}} \vee x_1^{t-1} \cdot x_2^{t-1} \vee x_1^{t-1} \cdot \overline{x_8^{t-1}}, x_3^t = x_3^{t-1} \cdot \overline{x_8^{t-1}} \vee x_1^{t-1} \cdot x_3^{t-1} \vee x_3^{t-1} \cdot \overline{x_8^{t-1}}, x_4^t = x_2^{t-1},\\
&x_5^t = \overline{x_4^{t-1}} \cdot x_5^{t-1} \cdot \overline{x_6^{t-1}} \cdot \overline{x_8^{t-1}} \vee x_5^{t-1} \cdot \overline{x_6^{t-1}} \cdot \overline{x_8^{t-1}} \cdot x_{10}^{t-1} \vee \overline{x_4^{t-1}} \cdot x_5^{t-1} \cdot \overline{x_8^{t-1}} \cdot x_{10}^{t-1} \vee \cdot \overline{x_4^{t-1}} \cdot x_5^{t-1} \cdot \overline{x_6^{t-1}} \cdot x_{10}^{t-1} \vee\\
&\vee x_5^{t-1} \cdot \overline{x_6^{t-1}} \cdot \overline{x_8^{t-1}} \cdot x_{11}^{t-1} \vee \overline{x_4^{t-1}} \cdot x_5^{t-1} \cdot \overline{x_8^{t-1}} \cdot x_{11}^{t-1} \vee \overline{x_4^{t-1}} \cdot x_5^{t-1} \cdot \overline{x_6^{t-1}} \cdot x_{11}^{t-1} \vee x_5^{t-1} \cdot \overline{x_8^{t-1}} \cdot x_{10}^{t-1} \cdot x_{11}^{t-1} \vee\\
&\vee x_5^{t-1} \cdot \overline{x_6^{t-1}} \cdot x_{10}^{t-1} \cdot x_{11}^{t-1} \vee \overline{x_4^{t-1}} \cdot x_5^{t-1} \cdot x_{10}^{t-1} \cdot x_{11}^{t-1} \vee \overline{x_4^{t-1}} \cdot \overline{x_6^{t-1}} \cdot \overline{x_8^{t-1}} \cdot x_{11}^{t-1} \vee \overline{x_6^{t-1}} \cdot \overline{x_8^{t-1}} \cdot x_{10}^{t-1} \cdot x_{11}^{t-1} \vee\\
&\vee \overline{x_4^{t-1}} \cdot \overline{x_8^{t-1}} \cdot x_{10}^{t-1} \cdot x_{11}^{t-1} \vee \overline{x_4^{t-1}} \cdot \overline{x_6^{t-1}} \cdot x_{10}^{t-1} \cdot x_{11}^{t-1} \vee \overline{x_4^{t-1}} \cdot \overline{x_6^{t-1}} \cdot \overline{x_8^{t-1}} \cdot x_{10}^{t-1},\\
&x_6^t = \overline{x_5^{t-1}} \cdot x_6^{t-1} \cdot \overline{x_{10}^{t-1}} \vee x_3^{t-1} \cdot x_6^{t-1} \cdot \overline{x_{10}^{t-1}} \vee x_3^{t-1} \cdot \overline{x_5^{t-1}} \cdot x_6^{t-1} \vee x_3^{t-1} \vee \overline{x_5^{t-1}} \cdot \overline{x_{10}^{t-1}},\\
&x_7^t = \overline{x_4^{t-1}} \cdot \overline{x_6^{t-1}} \cdot x_7^{t-1} \cdot \overline{x_8^{t-1}} \vee \overline{x_6^{t-1}} \cdot x_7^{t-1} \cdot \overline{x_8^{t-1}} \cdot x_{10}^{t-1} \vee \overline{x_4^{t-1}} \cdot x_7^{t-1} \cdot \overline{x_8^{t-1}} \cdot x_{10}^{t-1} \vee \overline{x_4^{t-1}} \cdot \overline{x_6^{t-1}} \cdot x_8^{t-1} \cdot x_{10}^{t-1} \vee\\
&\vee \overline{x_4^{t-1}} \cdot \overline{x_6^{t-1}} \cdot \overline{x_8^{t-1}} \cdot x_{10}^{t-1},\\
&x_8^t = \overline{x_5^{t-1}} \cdot x_6^{t-1} \cdot \overline{x_7^{t-1}} \cdot x_8^{t-1} \vee \overline{x_5^{t-1}} \cdot \overline{x_7^{t-1}} \cdot x_8^{t-1} \cdot x_9^{t-1} \vee x_6^{t-1} \cdot \overline{x_7^{t-1}} \cdot x_8^{t-1} \cdot x_9^{t-1} \vee \overline{x_5^{t-1}} \cdot x_6^{t-1} \cdot x_8^{t-1} \cdot x_9^{t-1} \vee\\
&\vee \overline{x_5^{t-1}} \cdot \overline{x_7^{t-1}} \cdot x_8^{t-1} \cdot \overline{x_{10}^{t-1}} \vee x_6^{t-1} \cdot \overline{x_7^{t-1}} \cdot x_8^{t-1} \cdot \overline{x_{10}^{t-1}} \vee \overline{x_5^{t-1}} \cdot x_6^{t-1} \cdot x_8^{t-1} \cdot \overline{x_{10}^{t-1}} \vee \overline{x_7^{t-1}} \cdot x_8^{t-1} \cdot x_9^{t-1} \cdot \overline{x_{10}^{t-1}} \vee\\
&\vee \overline{x_5^{t-1}} \cdot x_8^{t-1} \cdot x_9^{t-1} \cdot \overline{x_{10}^{t-1}} \vee x_6^{t-1} \cdot x_8^{t-1} \cdot x_9^{t-1} \cdot \overline{x_{10}^{t-1}} \vee \overline{x_5^{t-1}} \cdot x_6^{t-1} \cdot \overline{x_7^{t-1}} \cdot \overline{x_{10}^{t-1}} \vee \overline{x_5^{t-1}} \cdot \overline{x_7^{t-1}} \cdot x_9^{t-1} \cdot \overline{x_{10}^{t-1}} \vee\\
&\vee \overline{x_6^{t-1}} \cdot x_7^{t-1} \cdot \overline{x_9^{t-1}} \cdot x_{10}^{t-1} \vee \overline{x_5^{t-1}} \cdot x_6^{t-1} \cdot x_9^{t-1} \cdot \overline{x_{10}^{t-1}} \vee \overline{x_5^{t-1}} \cdot x_6^{t-1} \cdot \overline{x_7^{t-1}} \cdot x_9^{t-1},\\
&x_9^t = x_6^{t-1} \vee x_8^{t-1}, x_{10}^t = x_8^{t-1} \vee x_9^{t-1}, x_{11}^t = \overline{x_8^{t-1}} \cdot x_{10}^{t-1} \vee x_9^{t-1} \cdot x_{10}^{t-1} \vee \overline{x_8^{t-1}} \cdot x_9^{t-1}.
\end{aligned}
\tag{5}
$$

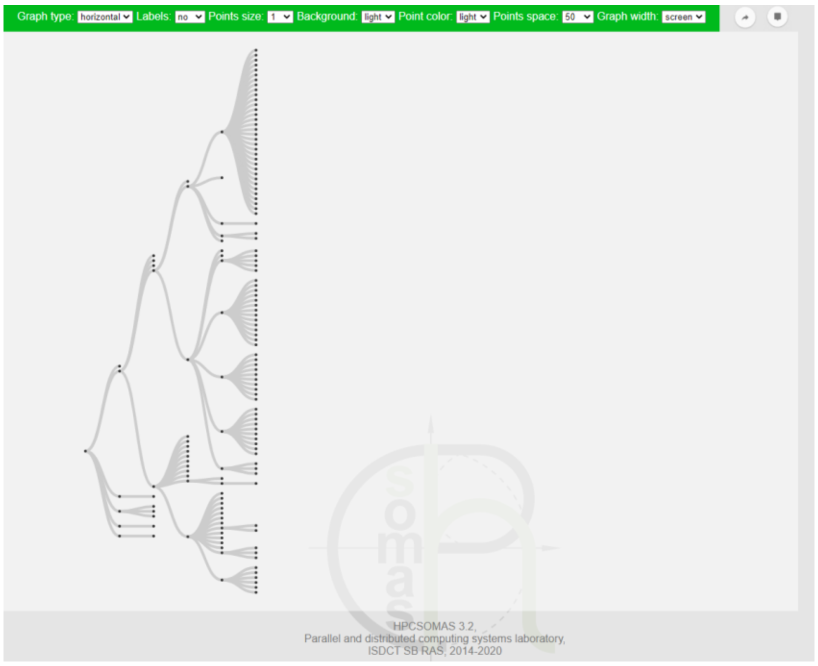
In (5), the indexes of the components x_i^0 of the Boolean state vector x in the time t=0 corresponds the following order of the gene regulator network [34, 35]: Cln3; SBF; MBF; Cln1,2; Sic1; Cln5,6; CDh1; Clb1,2; Mcm1; Cdc20; Swi5. The sign " $\cdot$ " denotes logical conjunction.

As a result, seven equilibrium states were found, including an isolated fixed point and six attractors (table 3). The presented results are consistent with those obtained in [34, 35]. The decimal code of the attractor state is calculated by the formula $\sum_{i=1}^{n} a_i * 2^{i-1}$.

For example, the decimal code of the attractor (01010000000) (table 3, no. 1) is calculated as 0*1+1*2+0*4+1*8+0*16+0*32+0*64+0*128+0*256+0*512+0*1024+0*2048=10. Attractor basins (*ABAS*) of the maximum radius *k* (*RAB*) are used for the visualization (figure 6-11). For this, the NPS T_2= (A_2^0= {*ABAS*, *Y_BAS*}; B_2^O = {*PDAB*}) is formulated. Using the resulting parameter *PDAB* (plotter data) of the NPS T_2 that stores in the user folder on the AMPMA resource, the developed by authors graphical microservice (Grapher) of the BCM-QABDS visualizes online a basin for the attractor no.1 from table 3 (figure 6).

Table 3. Equilibrium states for the BDS (4).

No	Equilibrium state	Basin size	Max radius (r)
1	01010000000	151	2
2	00101000000	7	2
3	00101010000	109	6
4	00000000000	7	1
5	00001000000	9	2
6	00000010000	1	Isolated
7	00001010000	1764	16


Graph type: horizontal Labels: no Points size: 1 Background: light Point color: light Points space: 50 Graph width: screen

HPCSOMAS 3.2,
Parallel and distributed computing systems laboratory,
ISDCT SB RAS, 2014-2020

Figure 6. The visualization of the basin of attractor number 1.

The Grapher interface allows the choice of visualization options (figure 6). For example, the first two options set the horizontal increasing radius of the basin tree and the absence of marks (the decimal attractor state code) at the graph vertices. Other basins of attractors (table 3) with labeled vertices of the graph are shown in figures 7-10. In the case of a large basin size, its structure is visualized using a radial tree (set by *Graph type* in the interface). Attractor number 7 from table 3 is shown in the form of a radial tree (Fig. 11). The attractor is marked in red. The service Grapher was developed using the static microservice composition because requirements to its functionality are unchanged.

Solving complex problems is carried out by formulating NPSs in the AMPMA interface with the subsequent creation and execution of the corresponding ensembles of microservices. In the presented example, NPSs T_1 and T_2 were used for finding the equilibrium states of a given BDS, determining whether they are attractors, finding attractor basins, and the post-processing of results for the visualization.

BCM-based approach to solving problems of qualitative analysis of BDS functioning on a finite time interval is achieved by constructing Boolean models of dynamic properties, checking their satisfiability by SAT or 2QBF solvers, and post-processing solutions. Unlike other SAT-oriented approaches, Boolean models contain constraints describing both dynamics of BDS and the specification of the dynamical property. BCM is a declarative method that provides the possibility of data parallelism by splitting the Boolean model and distributed solving the obtained independent subtasks. Thus, this approach allows us to significantly increase the dimension of the BDS state vector and the period of its functioning when solving the problems of the qualitative analysis of BDS in the HDCE.

Mathematical editors, converters, simplifiers, and generators are required for building Boolean models. Boolean model verification programs are sequential and parallel solvers that have different system requirements. The integration of the above heterogeneous software in the form of microservices is carried out based on presented HPCSOMAS-MSC tools for the development and execution of microservice compositions. The microservices are implemented based on RESTful style. This approach provides independence, replicability, the autonomy of used software, and its interaction through the lightweight message transfer mechanism.

Agents of the semantic P2P network perform decentralized asynchronous control of the microservice ensemble execution. This approach provides adaptability, reliability, and scalability of computations.

The openness of this agent network allows the allocation of additional resources in the case of its lack or failure.
HPCSOMAS-MSC tools for microservice composition development provide for creation microservices and agents both the class library and configurable ready-made patterns. Also, the system MSCDT for automation of these processes is provided. Thus, these tools are oriented on both professional programmers and end-users, specialists in a subject domain.

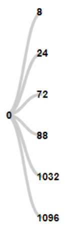


Figure 7. The basin of attractor number 4.

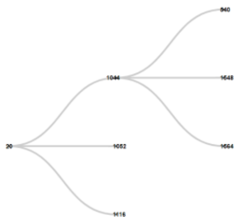


Figure 8. The basin of attractor number 2.

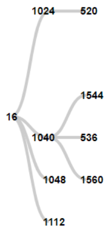


Figure 9. The basin of attractor number 5.

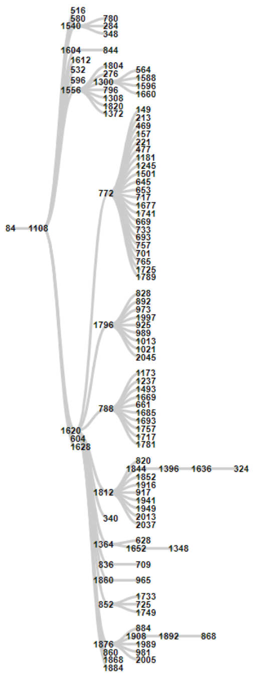
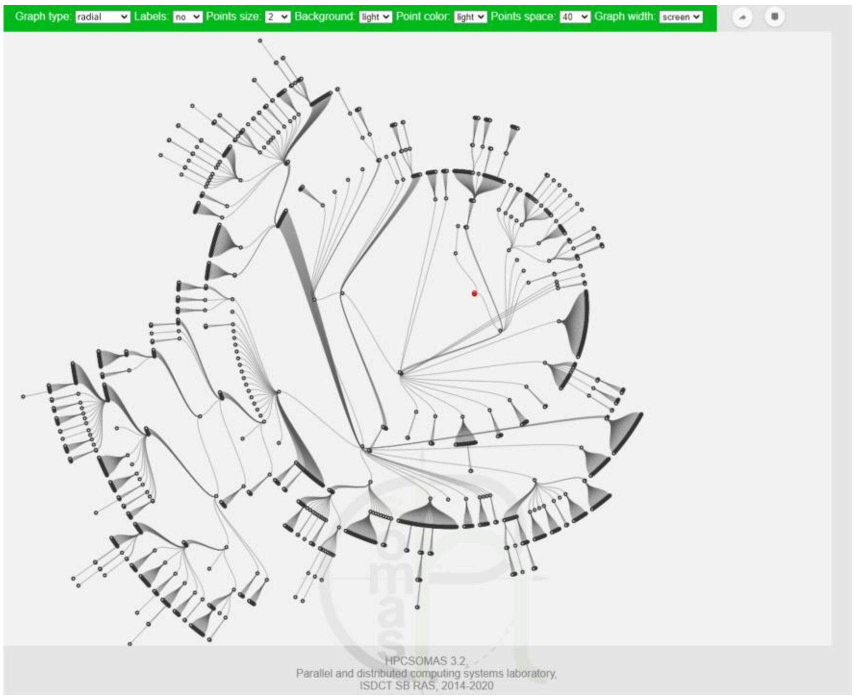


Figure 10. The basin of attractor number 3.



Graph type: radial Labels: no Points size: 2 Background: light Point color: light Points space: 40 Graph width: screen

HPCSOMAS 3.2,
Parallel and distributed computing systems laboratory,
ISDCT SB RAS, 2014-2020

Figure 11. The basin of attractor number 7.

5. Conclusion

Based on the HPCSOMAS-MSC framework, tools were developed to automate the construction and execution of composite services in the form of a composition of microservices and an ensemble of microservices based on respectively static or dynamic approaches. Composite service is performed in the first case based on hierarchical control, in the second - based on decentralized asynchronous control according to data readiness. The absence of a central control node increases the reliability of the calculations. Using a P2P agent network allows agents to be mobilized if it is necessary to attract additional resources due to their shortage or failure. These situations are handled in the behavioral model of agents [30]. The developed tools were tested on several problems from the previously mentioned research domain (for example, [11, 13, 14, 30]). As the development of research, the possibility of adapting behavioral models of agents oriented to different HPCSOMAS-MSC functioning modes is considered to provide a hybrid functioning, for example, launching an ensemble of microservices on a computational layer in the hierarchy of microservices composition.

Acknowledgments
The present investigation was supported by Russian Foundation of Basic Research, projects no. 18-07-00596. The authors would like to thank the Irkutsk Supercomputer Center of SB RAS for providing access to cluster computational resources.

References

[1] Tan W, Zhou M 2013 Business and Scientific Workflows: A Web service-oriented approach, *Wiley, IEEE Press*

[2] Huhns M N, Sing M P 2005 Service-oriented computing: key concepts and principles, *IEEE Internet Computing* **9** (1) pp 75–81

[3] Rao J and Su X 2005 A Survey of Automated Web Service composition methods *Semantic Web Services and Web Process Composition. SWSWPC*, LNCS vol 33872004 ed J Cardoso and A Sheth (Berlin, Heidelberg: Springer) pp 43-54

[4] Maamar Z and Wives L K 2010 Social networks and Web services-based systems. [Online]. Available: https://www.igi-global.com/chapter/social-networks-web-services-based/41252.

[5] Seheon Song and Seok-Won Lee 2013 A goal-driven approach for adaptive service composition using planning *Mathematical and Computer Modelling* **58** 261–273

[6] Sheng Q Z, Qiao X, Vasilakos A V, Szabo C, Bourne S and Xu X 2014 Web services composition: A decade's overview *Information Sciences* **280** 218–238

[7] Oparin G A, Bogdanova V G, Pashinin A A and Gorsky S A 2019 Microservice-oriented approach to automation of distributed scientific computations *Proc. of the 42st Int. Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2019, Opatija, Croatia, May 2019* pp 253-258

[8] Oparin G, Bogdanova V and Pashinin A 2019 Qualitative analysis of autonomous synchronous binary dynamic systems *MESA* **10**(3) 407-419

[9] Nemirovskii A A 1993 Several NP-hard problems arising in robust stability analysis *Mathematics of Control, Signals, and Systems* **6** 99 – 105

[10] Somov Ye I, Butyrin S, Oparin G A and Bogdanova V G 2016 Methods and software for computer-aided design of the spacecraft guidance, navigation and control systems *MESA* **7**(4) 613-624

[11] Bychkov I V , Oparin G A , Bogdanova V G, Pashinin A A and Gorsky S A 2017 Automation Development Framework of Scalable Scientific Web Applications Based on Subject Domain Knowledge *Parallel Computing Technologies. PaCT 2017. LNCS* vol 10421 ed V Malyshkin (Cham: Springer)

[12] Oparin G, Feoktistov A, Bogdanova V and Sidorov I 2016 Automation of multi-agent control for complex dynamic systems in heterogeneous computational network *AIP Conference Proceedings. ICNPAA-2016, July 5-8, La Rochelle, France* vol 1798(1) p 020117

[13] Bychkov I, Oparin G, Feoktistov A, Bogdanova V and Sidorov I 2017 The Service-Oriented Multiagent Approach to High-Performance Scientific Computing *Numerical Analysis and Its Applications. NAA 2016. LNCS* vol 10187 ed I Dimov I et. al (Cham: Springer) pp 261-268

[14] Oparin G, Bogdanova V, Gorsky S and Pashinin A 2019 The synthesis of stabilizing feedback for binary dynamic systems: a logical approach *MESA* **10**(3) 477-486

[15] Angel Lagares Lemos, Florian Daniel and Boualem Benatallah 2015 Web service composition: A survey of techniques and tools *ACM Computing Surveys* **48**(3) article 33

[16] Rodríguez G, Soria Á and Campo M 2016 AI-based Web Service Composition: A Review *IETE Technical Review* vol 33(4) pp 378-385

[17] Farnaghi M, Mansourian A 2018 Multi-Agent Planning for Automatic Geospatial Web Service Composition in Geoportals *ISPRS Int. J. Geo-Inf* **7** 404

[18] Hioual O and Boufaida Z 2011 An agent based architecture (using planning) for dynamic and

semantic web services composition in an ebxml context *Computer Science* [Online]. Available: http://arxiv.org/abs/1103.0632v1.

[19] Benatallah B, Dumas M, Sheng Q Z and Ngu A H H 2002 Declarative composition and peer-to-peer provisioning of dynamic Web services *Proc. 18th Int. Conf. on Data Engineering, San Jose, CA, USA, 2002* pp 297-308

[20] Shah Tejas R, Patel S V A Survey on issues and challenges of Web service development, composition, discovery *VNSGU journal of science and technology* **5**(1), 134 - 153

[21] Casati F, Ilnicki S, Jin L, Krishnamoorthy V and Shan M 2000 Adaptive and dynamic service composition in eFlow *Proc. of 12th Int. Conf. CAiSE 2000, Stockholm, Sweden, June 5–9, 2000 Proceedings CAISE* 13–31

[22] Pautasso C 2005 JOpera: An agile environment for web service composition with visual unit testing and refactoring. *Proc. of IEEE Symposium on Visual Languages and Human-Centric Computing* pp 311–313

[23] Hull D, Wolstencroft K, Stevens R et al. 2006 Taverna: a tool for building and running workflows of services *Nucleic Acids Research* **34** 729–732

[24] Deelman E, Vahi K, Rynge M, Juve G, Mayani R and Ferreira da Silva R 2016 Pegasus in the cloud: science automation through workflow technologies *IEEE Internet Computing* 20(1) 70-76

[25] Talia D. Workflow systems for science: concepts and tools. ISRN Software Engineering. Vol. 2013, Article ID 404525, 15 pages. http://dx.doi.org/10.1155/2013/404525

[26] Wolstencroft K., Haines R., Fellows D., Williams A., Withers D., Owen S., Goble C. (2013). The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. Nucleic Acids Research, 41(Web Server issue), W557–W561. http://doi.org/10.1093/nar/gkt328

[27] Oparin G A , Novopashin A P 2008 Boolean models and planning methods for parallel abstract programs *Autom Remote Control* **69** 1423–1432

[28] van der Aalst W, ter Hofstede A, Kiepuszewsk B et al. 2003 Workflow patterns *Distributed and Parallel Databases* **14** 5–51

[29] Oparin G A, Bogdanova V G, Pashinin A A and Gorsky S A 2018 Distributed solvers of applied problems based on microservices and agent networks *Proc. of the 41st Int. Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, May, 2018* pp 1415-1420

[30] Bychkov I, Oparin G, Bogdanova V and Pashinin A 2019 Intellectual technology for computation control in the package of applied microservices *Proc. of the 1st International Workshop on Information, Computation, and Control Systems for Distributed Environments, Irkutsk, Russia, July 8-9, 2019* pp 15-28

[31] Oparin G, Bogdanova V and Pashinin A 2019 Automation of microservices creation for qualitative analysis of binary dynamic systems *Proc. of the 1st International Workshop on Information, Computation, and Control Systems for Distributed Environments, Irkutsk, Russia, July 8-9, 2019* pp 88-98

[32] Irkutsk Supercomputer Center of SB RAS. [Online]. Available: http://hpc.icc.ru/.

[33] First VDS. [Online]. Available: https://firstvds.ru/.

[34] Li F, Long T, Lu Y, Ouyang Q and Tang C The yeast cell-cycle network is robustly designed *PNAS April 6, 2004* vol 101(14) pp 4781-4786

[35] Boolean network model of the control of the budding yeast cell cycle regulation. [Online]. Available: https://people.kth.se/~dubrova/BooleNet/budding_yeast.net