Distributed Streaming Data Processing in IoT Systems Using Multi-agent Software Architecture

Alexey Kovtunenko[1(✉)], Azat Bilyalov[2], and Sagit Valeev[1]

[1] Ufa State Aviation Technical University, Karl Marx st. 12, Ufa, Russia
askovtunenko@ugatu.su, vss2000@mail.ru
[2] Bashkir State Medical University, Teatralnaya st. 2a, Ufa, Russia
azat.bilyalov@gmail.com

Abstract. The problem of distributed storing and processing of streaming data in IoT systems is considered. A mathematical model and agent-based software architecture for distributed streaming data processing over heterogeneous computer network is offered. The software architecture determines the following features of IoT nodes software: structure of software components, models of interoperability, algorithms of resource management and also xml-based language which allows to descript distributed IoT applications. The offered architecture is implemented as a software framework ABSynth.

Keywords: Internet of Things · Streaming data · Software architecture
Distributed control system · Data stream management system

1 Introduction

Modern IoT systems include subsystems for processing and storing of streaming data (PSSD) as an integral part. Together they compose a large class of hardware-software complexes for the distributed collection, processing and storage of telemetry, monitoring of the state of technical and biological objects [1, 2]. Unlike traditional storage systems based on DBMS (database management systems), PSSD uses data stream management systems (DSMS) as a kernel and must provide a processing procedures priority higher than data storage procedures priority. They require faster access to data and less processing time. The specifics of the streaming data do not allow to use existing high-performance computing technologies in processing of streaming data [2].

In the article the agent-oriented approach to the organization of calculating process on the computer networks is discussed. It consists in establishment of special relations between program components and computing resources. Software complex components are presented by autonomous objects – software agents functioning in a certain environment – the agent platform. The agent platform is a software layer that provides a unified address space for software components, components interaction, and execution management.

Supported by Ufa State Aviation Technical University.

O. Galinina et al. (Eds.): NEW2AN 2018/ruSMART 2018, LNCS 11118, pp. 572–583, 2018.
https://doi.org/10.1007/978-3-030-01168-0_51

On the basis of the agent-oriented approach, a mathematical model of distributed data processing over on heterogeneous computer networks (HCN) with unpredictable loading (UL) is proposed. The problem of optimal resources allocation in heterogeneous computer network is formulated, and a multi-agent algorithm for dynamic control of computer network resources is discussed. The developed agent-oriented software architecture of distributed data processing systems is presented in the work.

The proposed approach and developed architectural and algorithmic solutions allow to design and create reliable distributed software for data processing in IoT systems based on HCN with UL.

2 State-of-the-Art Scientific Research in the Area of Streaming Data Processing

Streaming data processing is a modern approach, which regards the problem of data processing from a new point of view. Nowadays the efforts of researches are focused on the development of DSMS architectures in terms of providing the following functions: processing continuous queries (CQ) based on some query language or algebra for data streams [4] and providing a Quality of Service (QoS) based on some data stream model [5]. For example, Aurora [6] is a system for managing data streams for monitoring applications state. It uses simple dataflow diagrams to specify CQ and stream query algebra consisting of several primitive operators for expressing stream processing requirements [7].

Another aspect of scientific research in area of DSMS is complex event processing (CEP) systems: development of discrete-event models of different subject areas, formulating of logical rules for complex events generating. IBM's Active Middleware Technology (or AMiT) [8] is a tool that includes both a language and an efficient run-time execution engine, aimed at reducing the complexity of developing event-based applications. AMiT associates computations along with the definition of an event and uses event operators (following the ECA paradigm). AMiT is an example of a system that has tried to incorporate computations into an event specification.

Most of the considered systems represent a completed solution for event-based development, some of them even support distributed computations in common namespace, but they don't include a model of effectively computational resource allocation and don't allow dynamical management of computational resources.

3 Mathematical Models and Formulation of the Problem

The streaming data is a set of attributes, which has a different physical meaning (such as values of physical fields in different points, biometric indicators, simple and complex events et al.).

3.1 Model of Streaming Data Processing

The proposed mathematical model of PSSD is represented by the following tuple:

$$DPS = \langle A, P, C, I, M, V, T \rangle. \tag{1}$$

Here:

- A – set of attributes,
- P – set of computational procedures,
- C – a calculation relation that indicates which attributes are calculated by what procedure,

$$C = \{(p, a): p \text{ calculates } a;\ a \in A,\ p \in P\} \tag{2}$$

- I – a usage relation that indicates which attributes are used in calculations in each procedure, such that $C \cap I = \varnothing$,

$$I = \{(p, a): p \text{ uses } a;\ a \in A,\ p \in P\} \tag{3}$$

- $M{:}A \to \mathsf{N}\backslash\{0\}$ – a function that determines the memory consumption for storing attributes is such that *M(a)* is the amount of memory required to store the attribute a in bytes,
- $V{:}P \to \mathsf{N}\backslash\{0\}$ – a function that determines the computational complexity of the data processing task is such that *V(p)* is the computational volume of procedure p in the number of floating point operations,
- $T{:}P \to \mathsf{R}^{+}$ – the period of the procedure p in seconds.

Using the concept of the cut of the relation $Z{:}X \to Y$ with respect to the element x, as well as $Z(x) = \{y{:}\ (x, y) \in Z\}$ and the definition of the quotient set of Y determined by $\mathrm{Z}\ Y/Z = \{Z(x)\}_{x \in X}$, we can define the following families of sets:

- partition A/C of a set of attributes over a set of computational procedures according to a calculation, where *C(p)* is attributes whose values are calculating in procedure p,
- a family of subsets A/C, where *I(p)* is attributes which procedure p uses for calculations and reflects the algorithmic relationship between attributes.

To formulate the problem of optimal computing resources allocation in the PSSD, it is necessary to formulate a mathematical model for the resource consumption in computer networks taking into account the specificity of the streaming data.

3.2 Model of the Resource Consumption in Computer Networks

The computing resource of the network is traditionally considered to consist of three components:

- performance (CPU resource),
- memory,

– speed of network interactions.

The following tuple corresponds to the computing network.

$$N = \langle W, R, S \rangle \tag{4}$$

Here:

– W – set of computing nodes,
– $R = \{cpu, mem, net^{in}, net^{out}\}$ – set of computing resources,
– $S_r(w)$ – nonnegative function that defines the following parameters:
  - $S_{cpu}(w)$ flops – the node w maximum achievable performance;
  - $S_{mem}(w)$ bytes– the node w available RAM;
  - $S^{in}_{net}(w)$, $S^{out}_{net}(w)$ bytes/second – the maximum achievable network rate for the incoming and outgoing traffic of the node w, respectively.

Various ways of specifying a function S lead to models of different network types.

3.3 Formulation of the Optimal Resource Allocation Problem

The model of the PSSD based on the computer network includes resource model of the computer network N, model of data processing DPS and the relation D describing system deployment in network.

$$IS = \langle N, DPS, D \rangle \tag{5}$$

Here:

– N – the computational network,
– DPS – the PSSD model,
– D – the deployment relation that shows $\forall p \in P$ on what node it physically deployed

$$D = \{(w, p): w \text{ executes } p;\ w \in W,\ p \in P\}. \tag{6}$$

Let $B: W \times R \to \mathsf{R}^+$ – function that shows $\forall w \in W$ amount of the involved resource. The introduced notation makes it possible to formulate the problem of optimal allocation of computational resources in PSSD as the problem of combinatorial optimization. It is necessary to find such deployment relation D^*, that gives maximum to predefined efficiency function F.

$$D * = \arg \max_{D \subset W \times P} F(IS\langle D \rangle) \tag{7}$$

The following constraints apply.

$$B_r(w) \le S_r(w) \tag{8}$$

4 Agent-Based Software Architecture for PSSD Systems

In line to proposed mathematical models the agent-based software architecture is developed [9]. It assumes software consisting of next structure components:

– the set of **target agents**, where each one is software entity that implements some data processing procedure;
– the set of processing **attributes** implemented as in-memory database (IMDB), that provides distributed storing of streaming data and a low-latency access to them.

To make a flexible, survivable and scalable software the agent approach is proposed, which means using a software agent as a basic software entity. It allows us hereinafter to talk about agent-based software architecture and implies the presence of some additional elements in the system, such as

– **agent platform** – a set of software components that supports the life cycle of target agents, interaction between them in the unified namespace, also the system event model
– **agent container** – a software environment that provides an access to the resources of each computing node and executes of software agents.

Thus, the streaming data processing model within the software architecture proposed above can be represented by the following structural diagram (Fig. 1).

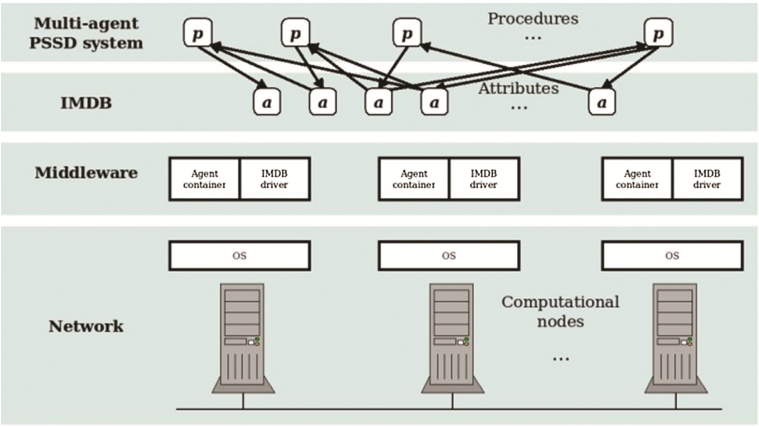

Fig. 1. Software architecture for distributed streaming data processing

A unified storage space for streaming data is provided by a distributed cached in-memory storage that provides fast and transparent access to attribute values regardless of their physical location. The agent platform is a unified distributed execution space for data processing procedures.

4.1 The Target Agent's Architectural Features

The **target agent** is the basic software entity in the implementation of the data processing model within the proposed architecture [10]. It includes the following components:

- data processing procedure $p \in P$;
- local cache of used attributes $I(p) \subset A$;
- local cache of calculated attributes $C(p) \subset A$:
- mechanism for the management messages receiving and executing;
- mechanism for the agent initialization and self-destruction;
- agent's state.

The activity diagram of the **target agent** shows the process of the agent's functioning after creation (Fig. 2).

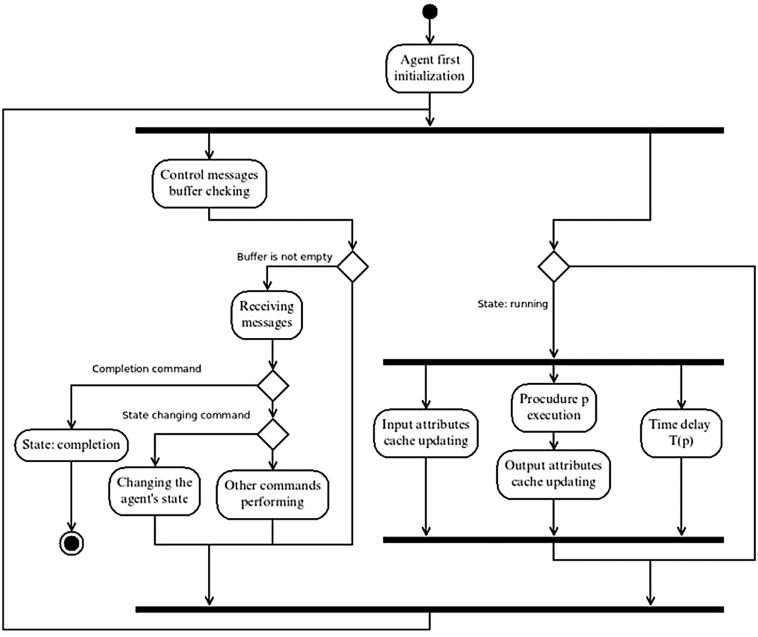

Fig. 2. The fragment of activity diagram of the target agent

In the process of distributed data processing, the **target agent** can be in the following states.

- "Created" – the initial state in the agent's lifecycle.
- "Initialized" – the calculated attributes are set to the initial values.
- "Running" – the agent's operating state when the attributes are recalculating in cyclic mode. "Holding" the agent's operating state when recalculating of attributes is temporary suspended.

– "Completion" is the final state in the agent's lifecycle (preparation for self-destruction, completion of all recording processes).

Changes of the **target agent's** state occurs either according to the stage of the lifecycle, or under the influence of management commands.

4.2 Management Within the Agent-Based Software Architecture

The management of the streaming data processing, as well as the allocation of resources within the agent-oriented software platform is carried out through special **service agents** (Fig. 3):

– **global system manager** – unique (if there is no global manager duplication mode) for each PSSD system;
– **local node manager** – one for each computational core.

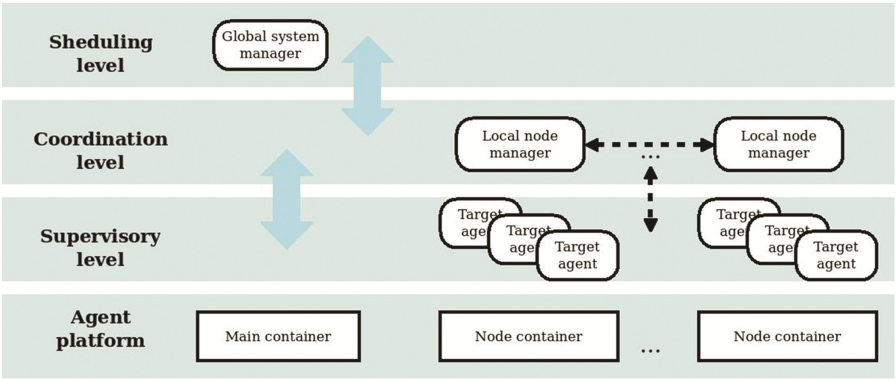

Fig. 3. Hybrid model of execute and resource allocation management in the PSSD system

Execution management implements a hierarchical model (wide solid arrows in the Fig. 3), which means interactions only between the management levels, and never – inside. The executive level is represented by the **target agents** that can start and stop execution and destroy themselves at the commands, received from higher management levels. The coordination level is represented by **local managers** who pass the commands of the **global manager** to the **target agents** of their node, within the management model. The planning level is represented by the **global manager**, as well as by the operator or supreme planning system [10].

Resource management is carried out within the hybrid model (thin dotted arrows in the Fig. 3), where there is a two-level hierarchical organization between **target agents** and **local managers**, as well as a multi-agent peer-to-peer model between **local managers**. The **service agents** interact with the **target agents** according to the specific protocol, proposed within the software architecture. It is based on the ACL [11] language of the FIPA standard and implies communication acts in the multi-agent software system within the following ontologies.

– **Management ontology** – managing commands within the hierarchical model (wide solid arrows at the Fig. 3).
– **Resource allocation ontology** – commands and data related to procedures of a computational resources allocation, such as: estimation of available resources for every node, heuristic algorithm for load balancing etc.

ABSynth allows to extend itself according to developer's needs by adding ontologies or supplementing of the existing ones.

4.3 Tasks Specification Description Language

For the formal description of multi-agent distributed PSSD systems the task specifications description language (TSDL) is offered. It is based on XML notation and consists of the following set of elements (in this section, the term "attribute" should be understood as an XML attribute – named string parameter of XML element which is written inside the angle brackets of the opening tag after its name):

– **<model > … </model >** – root element containing all other elements inside. It can have attributes allowing to specify some default parameters of PSSD, for example, period and executing node (attributes "period" and "container"), name and developer of the system (attributes "name" and "author").
– **<agent > … </agent >** – The element that defines the creation of the **target agent**. It contains all of the parameters and settings of the **target agent** as XML-attributes and subelements. Attributes of this element allow to set up for each target the following parameters: a unique name of the agent (attribute "name"), type of the agent (parameter "class"), period of recalculating values of PSSD attributes in milliseconds (attribute "period"), and placement (name of the container or computational node, where agent is physically deployed – attribute "container"). Subelements of this element allow to specify for each target agent a set of other target agents, whose PSSD attributes are used in its calculations (elements **< input name=""/>),** also a set of additional string parameters that need to be passed to it after creation, like initial values, settings, etc. (elements **< parameter name="" > … </parameter >).**

Also there are some elements that simplify writing of TSDL code and make it more readable.

– **<include > … </include >** – for complex multi-file models delineates an optional section where files with other model fragments can be specified.
– **<constant > … </constant >** – associates string content of an element to identifier defined as the attribute "name". Everywhere further identifier enclosed in braces will be interpreted as a string associated with it.

TSDL allows development of distributed PSSD systems specifications using text editors with syntax highlighting as simple as using special XML-based visual diagrams editors.

5 Software Implementation of Multi-agent Architecture for Distributed PSSD Systems and Examples of Using

Developed software architecture is implemented as a software framework **ABSynth**, based on java agent development environment (**JADE**) framework and inherits all its abilities and functionality. In addition to them **ABSynth** includes IMDB **Hazelcast** as a transport level for streaming data sharing.

Basic software entity of **ABSynth** in accordance with architecture is abstract java-class **BasicUnit** – implementation of **target agent**. **BasicUnit** extends the class **jade.Agent** from the JADE library and contains the abstract procedure **MainAction()** which is called with predefined period. Each data processing **target agent** should be implemented as a child class extending **BasicUnit**. The program code performing data processing should be placed into overridden **MainAction()** function. It can operate only with local cache of streaming data as with the class fields.

If one needs it is possible to create different agents (by extending the **BasicUnit** class) to perform one-type and diverse operations over the streaming data, such as, for example, assembling of raw data by sensors poll, signal filtration, indirect measuring (calculating not directly measureable parameters of business process), calculating of macroscopic parameters (some integral estimation, the sliding average), complex signals generation (events, automatic decisions) et cetera.

Let's consider some ways of possible using of the developed architectural and software solutions.

5.1 Networked Control Systems (NCS)

The most widespread scope of use of IoT systems is remote control of large-scale technical systems. The use of a heterogeneous computer network as part of automatic control systems has given rise to a new direction in control theory – networked control systems (NCS). The classical definition of NCS can be as follows: when a traditional feedback control system is closed via a communication channel, which may be shared with other nodes outside the control system, then the control system is called an NCS [12]. From the most general point of view the structure of NCS may be depicted as showed at the Fig. 4. A distinctive feature of the NCS as a software-hardware complex is that, the data, circulating between the network nodes, are streaming data. So, the implementation of control models and algorithms needs particular approach [13, 14].

The software layer of the NSC's technical equipment may be considered as a distributed PSSD, where the digital signals are streaming data, logical signals are complex events and control algorithms are streaming data processing procedures [15].

The example of the problem of semi-natural testing of the electronic control unit for gas-turbine engine was considered [16].

The software layer of the stand is based on ABSynth framework and contains the following types of target agents [17].

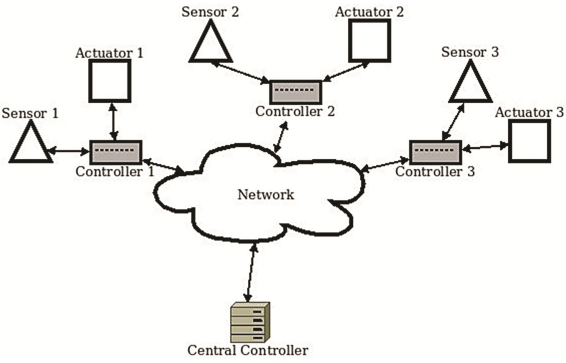

Fig. 4. The common schema of NCS

- Agent – element-wise computational model of the gas-turbin engine.
- Agents – drivers for all hardware modules (interfaces for the aircraft native communication buses, agents for simulation of the mechanical and electrical faults and others).
- Agents – simulation models for the other modules of aircraft automatics (for example the throttle).
- Agents – simulation models for flight dynamics of the aircraft (for example, geometrical position of the aircraft and its parts relative to the ground: the roll angles, pitch and yawing, the geostationary coordinates of the mass center).
- Agents – simulation models for the environment in flight conditions (the input air pressure of the engine, air resistance, the lift force depending on the geometrical position of the aircraft).
- Agents – elements of the graphical user interface (GUI), which allow the researchers to monitor the testing process (depicts all signals in the human-understandable form like numerical tables or time diagrams).

Using the ABSynth platform together with the classes presented above allows to perform the semi-natural testing as well as the full simulation (without natural parts) with involving of different types of computer networks. Different fragments of the model can be developed by separate research teams, copyrighted, and executed using computational equipment in different institutions.

For example, let's consider a simulation of starting and thrust-increment regime for the gas-turbine engine in conditions of different pressure amount of the input air. At the Fig. 5 time diagram of the rotation speed of the low-pressure turbine is represented.

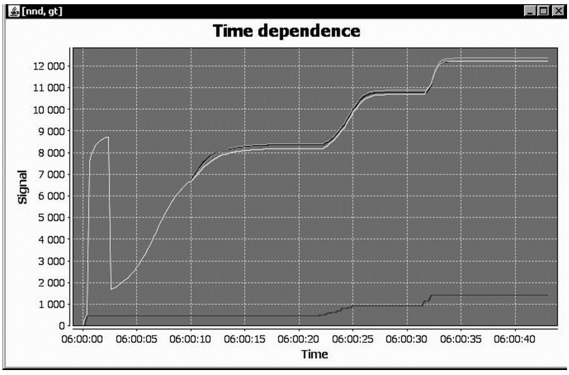

Fig. 5. Simulating of the low-pressure turbine rotation speed

Each instance of the engine executed autonomously at its own node of the computational network. Mechanisms of the ABSynth framework allowed to perform the simulation in real-time.

6 Conclusion

The main results of the work are presented below.

- The overview of state-of-the-art scientific research in the area of streaming data processing.
- The mathematical model of streaming data processing.
- The mathematical model of the resource consumption in computer networks.
- The formulation of the optimal resource allocation problem in heterogeneous networks
- Agent-based software architecture for distributed processing and storing of streaming data including the structure of components and TSDL, which is the language to describe multi-agent applications.
- The ABSynth platform is based on software agents technology, which implements all the developed models and architectural solutions.

References

1. Khakimov, A., Muthanna, A., Kirichek, R., Koucheryavy, A., Muthanna, M.S.A.: Investigation of methods for remote control IoT-devices based on cloud platforms and different interaction protocols. In: Proceedings of the 2017 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Moscow, St. Petersburg (2017)
2. Ateya, A.A., Muthanna, A., Gudkova, I., Abuarqoub, A., Vybornova, A., Koucheryavy, A.: Development of intelligent core network for tactile internet and future smart systems. J. Sens. Actuator Netw. **7**(1) (2018)
3. Ramakrishnan, R.: Database Management Systems. WCB/McGraw-Hill (1998)

4. Soulé, R., et al.: A unified semantics for stream processing languages (extended). Technical report 2010-924, New York University (2010)
5. Chakravarthy, S., Jiang, Q.: Stream Data Processing: A Quality of Service Perspective Modeling, Scheduling, Load Shedding, and Complex Event Processing. Springer, Heidelberg (2009)
6. Abadi, D., et al.: Aurora: a new model and architecture for data stream management. VLDB J. **12**(2), 120–139 (2003)
7. Zdonik, S.B., Stonebraker, M., Cherniack, M., Çetintemel, U., Balazinska, M., Balakrishnan, H.: The aurora and medusa projects. IEEE Data Eng. Bull. **26**(1), 3–10 (2003)
8. Adi, A., Etzion, O.: AMiT – the situation manager. VLDB J. **13**(2), 177–203 (2004)
9. Valeev, S.S., Maslennikov, V.A., Kovtunenko, A.S.: Design of the middleware based on agent-oriented technologies for the automated control systems of complex technical objects. In: Proceedings of 6th International Conference "Parallel Computing and Control Problems" vol. 1, Moscow (2012)
10. Kovtunenko, A.S., Valeev, S.S., Maslennikov, V.A.: Agent-oriented software architecture for distributed processing and storing of streaming data. In: Proceedings of the 2nd International Conference on Intelligent Technologies for Information Processing and Management (ITIPM 2014), Ufa (2014)
11. FIPA ACL Message Structure Specification. Foundation for Intelligent Physical Agents (2002). http://www.fipa.org/specs/fipa00061/SC00061G.html
12. Zhao, Y.B., Sun, X.M., Zhang, J., Shi, P.: Networked control systems: the communication basics and control methodologies. Math. Prob. Eng. (2015)
13. Lin, H., et al.: Estimation and Control for Networked Systems with Packet Losses without Acknowledgement, Studies in Systems, Decision and Control, vol. 77. Springer (2017)
14. Volkov, A., Khakimov, A., Muthanna, A., Kirichek, R., Vladyko, A., Koucheryavy, A.: Interaction of the IoT traffic generated by a smart city segment with SDN core network. In: Koucheryavy, Y., Mamatas, L., Matta, I., Ometov, A., Papadimitriou, P. (eds.) WWIC 2017. LNCS, vol. 10372, pp. 115–126. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-61382-6_10
15. Kovtunenko, A.S., Maslennikov, V.A.: Creation of distributed control information systems on the basis of agent-oriented approach. In: XII All-Russian Conference on Problems of Management, Moscow (2014)
16. Valeev, S.S., Zagitova, A.I., Kovtunenko, A.S.: Simulation of the gas-turbine aviation engine under flight conditions using the ABSynth multiagent platform. In: Proceedings of the 3rd International Conference on Information Technologies for Intelligent Decision Making Support ITIDS 2015, Ufa (2015)
17. Kovtunenko, A.S., Valeev, S.S., Maslennikov, V.A.: A multi-agent platform for distributed real-time processing. Nat. Tech. Sci. **2**(64) (2013)