# Control system for multi-agent groups of heterogeneous sensors

Andrey Kulikov [1], Alexandr Timoshenko [2], Alexandr Zhukov [1,3,4,5] and Igor Kartsan [3,6,7,8]

[1] *MIREA - Russian Technological University, 78, Vernadskogo Av., Moscow, 119454, Russia*
[2] *JSC "Radio Engineering Institute named after Academician A.L. Mints", 8 Marta Str., 10/1, Moscow, 127083, Russia*
[3] *FGBNU "Expert and Analytical Center", Talalikhina Str., 33, Building 4, Moscow, Russia*
[4] *Institute of Astronomy of the Russian Academy of Sciences, 48, Pyatnitskaya Str., Moscow, Russia*
[5] *Joint Stock Company "Special Research of Moscow Power Engineering Institute", 14, Krasnokazarmennaya Str., Moscow, Russia*
[6] *Marine Hydrophysical Institute, Russian Academy of Sciences», 2, Kapitanskaya Str., Sevastopol, Russia*
[7] *Reshetnev Siberian State University of Science and Technology, 31, Krasnoyarsky Rabochy Av., Krasnoyarsk, Russia*
[8] *Sevastopol State University, University Str. 33, Sevastopol, Russia*

**Abstract**

The results of the analysis of experience in creating multi-agent sensor group control systems used to solve the problems of monitoring complex environments under conditions of dynamic changes in the composition of the group and the action of destabilizing factors in real time are presented. It was shown that in forming requirements to prospective monitoring systems it is necessary to ensure a minimum feedback time from sensors and sensors, the sensors mean a cybernetic device with a transceiver for receiving commands and transmitting information, for adequate management commands, as well as the presence of a decision-making assistance system for operators when working in manual or semi-automatic modes. According to the results of the analysis, it was concluded about the importance of using SOA architecture, service-oriented, in the design of software to ensure the possibility of flexible connection of sensors and systems, as well as the implementation of such important principles as scalability, alignment of organizational and structural interactions, and analysis of information received in real time and post analysis to implement decision support, the basic decisions in creating hardware.

**Keywords**

Control systems, software architectures, technical requirements

## 1. Introduction

The development of a management system for multi-agent sensor groups used to solve problems of monitoring complex environments under conditions of dynamic changes in group composition and the action of destabilizing factors in real time is a science-intensive process because within the big task there are the following, subproblems:

- scalability;
- task distribution in a group of heterogeneous sensors;
- decision support;
- support for heterogeneous action reconciliation, etc.

Proceedings of MIP Computing-V 2022: V International Scientific Workshop on Modeling, Information Processing and Computing, January 25, 2022, Krasnoyarsk, Russia
EMAIL: science.andrey.kulikov@gmail.com (A Kulikov); u567ku78@gmail.com (A Timoshenko); aozhukov@mail.ru (A Zhukov); kartsan2003@mail.ru (I Kartsan)
ORCID: 0000-0001-6143-4986 (A Kulikov); 0000-0002-5122-3752 (A Zhukov); 0000-0003-1833-4036 (I Kartsan)

Given the actualization of the issue of automation of industrial processes in the conditions of the fourth industrial revolution, as well as the interest of various departments, the experience of creating such systems in terms of software and hardware implementation is highly relevant.

As part of the issue of creating heterogeneous sensor control systems, the following major projects have been implemented: CLARATy, METERON, RoboEarth, ROS.

However, all of them are implemented on the following principles of control systems implementation, namely:

- Distributed object architecture (DOA),
- Component-based architecture (CBA).
- Service Oriented Architecture (SOA).
- Robotics technology components (RTC)
- Collaborative architecture for unmanned systems (JAUS).

Distributed object architecture (DOA) concepts, are the result of a fusion of object-oriented design methods for distributed computing systems.

Advantages of the architecture, detailed description of digital objects.

The disadvantages of the architecture, in the implementation of the proposed system, the number of sensors of different kinds can be prohibitive and the description of each sensor in the system is not appropriate in terms of real-time operation of the system and code flexibility.

Thus, this architecture is not suitable for the implementation of the system under development.

CBA, is an evolutionary development of DOA approach in program design. The key feature is to create relationships between objects for specific scenarios.

Advantages of the architecture, detailed description of digital objects and relationships between objects for specific scenarios.

Disadvantages of the architecture, when implementing the proposed system, the number of sensors of different kinds can be prohibitive and the description of each sensor and the relationship between sensors in the system is not appropriate in terms of real-time operation of the system and code flexibility.

Thus, this architecture is not suitable for the implementation of the system under development.

Service-oriented architecture is a modular approach to software development, based on the use of distributed, loosely coupled replaceable components, equipped with standardized interfaces for interaction on standardized protocols.

The advantages of the architecture, the easy ability to improve the system. The detailed description of the interaction protocols, promotes the different structures and modules of the system to address other modules of the system. Due to flexibility, the possibility of support of past functionality in new versions of the system is realized.

Disadvantages of architecture, it is important factor of check of input data that they were in accordance with designed protocols.

Thus, this variant of architecture is suitable for the implementation of the system under development.

The JAUS architecture is based on components of a specific functional purpose, the messages that these components exchange with each other during operation, and the interfaces for integrating and connecting new components to the system. Each JAUS object is capable of responding to external requests (e.g., from control modules) and independently determining what it does, what data it possesses, and how it interacts with its environment.

JAUS is a set of documents describing data formats and methods for high-level interaction of software components. This architecture was originally intended for building ground-based robots, but is now focused on autonomous vehicles of any type (air, land, over- and underwater, military and civilian) and guarantees a unified scheme of remote control of an arbitrary JAUS model.

The main requirement for building a JAUS system is that all messages exchanged between components must be JAUS-compliant. And the messages themselves are considered the only way of communication between the components. The structure of these messages is designed in such a way as to minimize the bandwidth requirements of the communication line.

JAUS interoperability of devices must be ensured at three levels: between subsystems (robot - robot, robot - controller), between nodes (onboard radar - onboard controller) and between individual components.

Advantages of the architecture, the detailed and regulated structure allows to describe clearly the protocols of interaction, to regulate the structure of subordination. SOA like architecture.

Disadvantages of the architecture, due to the rigid description, rapid scaling of the system will be difficult and will impose additional time costs when working in real time.

Thus, the JAUS architecture is suitable when implementing the intended system.

## 2. Analysis of projects that use system design architectures

A study of control system construction principles has shown that different paradigms have different characteristics and properties that make them suitable for different distributed applications. DOA is based object, which is suitable for the lower levels, where developers require high performance, even if it requires a high level of concurrency control in the interaction of multiple objects. Instead, CBA and related middleware is more suitable for the middle levels, where the goal is to develop autonomous components that can be exchanged and composed based on applications. Finally, SOA is useful for developing loosely coupled architectures where interacting objects can be accessed without prior knowledge. The architecture of JAUS, is a promising one with a clear administrative monitoring of system usage and messaging protocol across all devices. SOA and JAUS type architectures can be useful to design the system for which the research is done. Further there is an analysis of real projects designed on SOA and JAUS architectures.

A division of the government agency NASA. The center has been involved in many NASA missions, leading in astrobiology, small spacecraft, robotic lunar exploration, technology for the Constellation Program, the search for habitable planets, supercomputers, smart and adaptive systems, thermal protection, and aerial astronomy. Ames is the center of several key NASA science missions (Kepler Missions, LCROSS, SOFIA, LADEE). Significant contributions have been made to the Exploration Project (Orion spacecraft and Ares I boosters). The CLARAty project was developed as part of this center Figure 1.

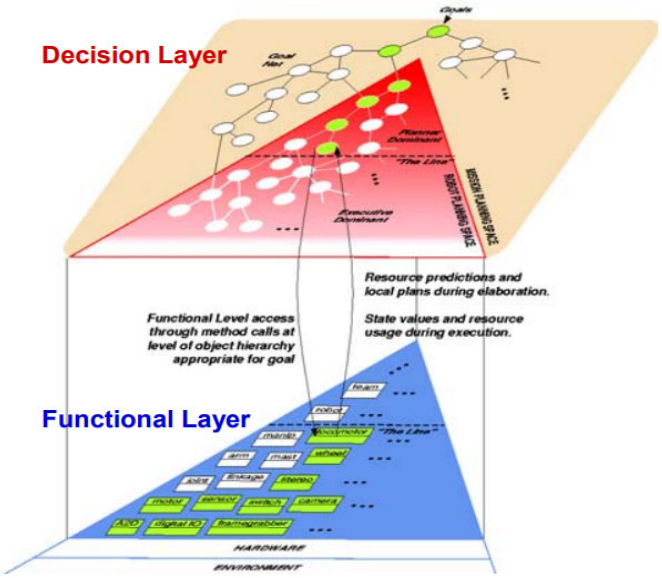


**Figure 1:** An abstract view of CLARATy and its inner sublevels

The CLARAty architecture has two different levels: the functional level and the decision-making level. The functional layer uses an object-oriented decomposition of the system and uses a number of well-known SOA design patterns to achieve reusable and extensible components. These components define the interface and provide basic system functionality that can be adapted to various real-world or simulated robots. The interface provides both low- and medium-level autonomy. The decision-making layer links the planning and execution system. The decision-making layer reason globally about the intended goal, system resources, and the state of the system and its environment. The decision level

uses a declarative model, while the functional level uses a procedural model. Because the functional level provides adaptations to the physical or simulated system, all of the specific model information is combined in the system adaptations. The decision-making layer obtains this information by querying the functional layer for predictive resource use, state updates, and model information.

METERON is a project being developed jointly with the space agencies of the United States, the European Union and Russia. The project implemented a large infrastructure to control the final Mars rovers and access to the obtained data in near real time from any command post of the countries participating in this project Figure 2.

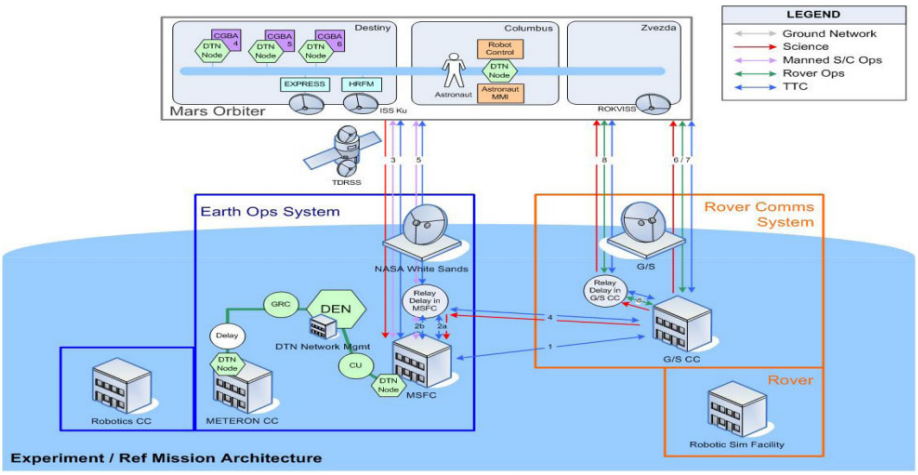


**Figure 2:** The shape of the METERON project infrastructure

The project was implemented according to the DOA design paradigm. The control delay of the robotic vehicle used in the METERON project. From the operator to the robotic means was 25-50 ms, considering that the robotic means itself communicated with the METERON system infrastructure via a Wi-Fi modem installed on the robot via Beagleboard, in conditions close to the real one. In real conditions, the real-time latency could be up to 102 ms.

RoboEarth, the ideology of the creation of this project, lies in the paradigm of software design in SOA architecture Figure 3.

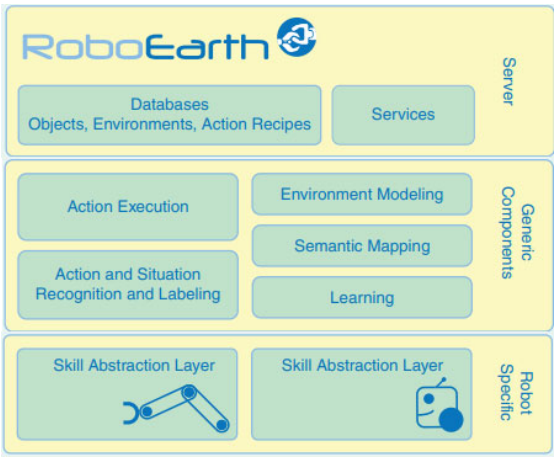


**Figure 3:** Three interconnected levels of the RoboEarth system

RoboEarth is realized on the basis of a three-level architecture. The core of this architecture is the server layer, which contains the RoboEarth database. It stores globally the model, including reusable, object information (e.g., images, point cloud, models), environments (e.g., maps and object locations), and actions (e.g., scripts and functions) associated with semantic information (e.g., object properties), and provides the underlying logic for web services. The database and database services are accessed through common web interfaces. The second layer implements universal components, which are part

of the local control program of the robot. Their main purpose is to allow the robot to interpret its actions in RoboEarth. Additional components extend the robot's analysis, modeling and learning functionality, which closes the loop from the robot to the first level. The third level implements skills and provides a general interface to specific hardware-dependent functions of the robot through the skill abstraction level.

A robotics operating system (ROS) developed by a Stanford research team and further developed by Willow Garage. The project is implemented as Open Source, can run on Linux-oriented hardware with strong computing power. It integrates with Gazebo software for research visualization and computer simulation of robots which are equipped with ROS.

When ROS runs, a "graph" - a point-to-point (peer-to-peer network) of processes is built, which communicate with each other through the ROS infrastructure.

ROS implements several different styles of communication: synchronous (RPC-style) communication between services, asynchronous data flows through Topics, data storage on the Parameter Server. ROS is not a real-time system, although it is possible to integrate ROS with real-time code.

ROS works on the SOA paradigm and evolves through research done by international scientific and amateur teams by releasing new libraries to implement this or that functionality.

DREAMachine (Defect / Test Reduction Empowered by Analytics and Machine Learning) is based on the application of machine learning methods that allow to quickly analyze huge amounts of information, recognize complex dependencies and predict the required output information for decision-making based on the available data.

DREAMachine uses a modular architecture that is extremely similar in design to SOA to achieve greater predictive accuracy and insight into system-level processes, as well as flexible application of developed algorithms to solve similar problems.

DREAMachine testing (Raytheon experts have set a precedent and substantiated the application of machine learning methods) showed the following results:

- teacher-assisted learning methods provided up to 99% accuracy in predicting failures of both components and the system being analyzed as a whole;
- the methods of learning without a teacher identified areas of redundancy in the testing process, providing opportunities to optimize this process;
- the proposed process optimization can increase production capacity by 40%.

## 3. System hardware and software requirements

Due to the fact that the management is planned in real-time, it is necessary to design the node of storage and processing of large and ultra-large amounts of information using server nodes on the software shells Apache Flink and Apache Storm. In addition, since there are heterogeneous sensors within the system and their scalability can be drastic, the hardware under development should, be equipped with support for Docker container technology or others. Thus, according to the results of the analysis and past research within the subject, the system under development, should be Linux-oriented and have basic characteristics.

## 4. Conclusion

Thus, the paper analyzed the approaches and methods of designing flexible and high-loaded systems. When developing a heterogeneous sensor control system, based on the reconfiguration of the composition and structure of the system in conditions of destabilizing factors, it is important to focus on SOA (Service-Oriented Architecture) programming architecture. Because, this architecture provides flexibility in dealing with heterogeneous robots and sensors, as proved by the projects of international teams. Also, the testing of this method of system design by Rayethon scientists led to the conclusion that the use of SOA will increase the performance of the analytics system in real time by 40%.

## 5. Acknowledgements

This work was carried out within the framework of the state assignment of the Ministry of Education and Science of the Russian Federation on "Conceptual modeling of the information and educational environment of human capital reproduction in a digital economy" 121102600069-2.

This work was performed within the framework of the state assignment of the Ministry of Education and Science of Russia on the topic "Development of new methods of autonomous navigation of spacecraft in outer space" 121102600068-5.

This study was supported by the Russian Federation State Task No 0555-2021-0005.

## 6. References

[1] T. Zikratova, The method of group control in multi-agent robotic systems under the influence of destabilizing factors, Proc. of Telecom. Universities 7(3) (2021) 92-100. doi:10.31854/1813-324X2021-7-3-92-100.

[2] V. I. Gorodetsky, Behavioral model for cyber-physical system and group control: the basic concepts, IZVESTIYA SFedU. ENGINEERING SCIENCES 1(203) (2019) 144-162. doi:10.23683/2311-3103-2019-1-144-162.

[3] A. O. Zhukov, A. K. Kulikov, I. N. Kartsan, Optimization of the control algorithm for heterogeneous robotic agricultural monitoring tools, IOP Conference Series: Earth and Environmental Science 839(3) 032039 (2021). doi: 10.1088/1755-1315/839/3/032039.

[4] S. V. Efremova, I. N. Kartsan, A. O. Zhukov, An ordered ranking multi-attributive model for decision-making systems with attributes of control systems software, IOP Conference Series: Materials Science and Engineering, 1047(1) 012068 (2021). doi: 10.1088/1757-899X/1047/1/012068.

[5] I. N. Kartsan, S. V. Efremova, V. V. Khrapunova, M. I. Tolstopiatov, Choice of optimal multiversion software for a small satellite ground-based control and command complex, IOP Conference Series: Materials Science and Engineering 450(2) 022015 (2018). doi: 10.1088/1757-899X/450/2/022015.

[6] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: a review from the swarm engineering perspective, Swarm Intelligence 7 1-41 (2013). doi:10.1007/s11721-012-0075-2.

[7] A. S. Kremlev, S. A. Kolyubin, S. A. Vrazhevsky, Autonomous multi-agent "robot-guide" system to solve area monitoring problems, Journal of Instrument Engineering (Izvestiya vysshikh uchebnykh zavedeniy. Priborostroenie) 56(4) (2013) 61-65.

[8] I. A. Kalyaev, A. R. Gaiduk, S. G. Kapustyan, Collective control models and algorithms in groups of robots. Moscow, FIZMATLIT Publ. (2009) 280.

[9] A. V. Masloboev, V. A. Putilov, Development and implementation of mobile agent security control mechanisms in the distributed multi-agent information systems, Vestnik of MSTU 13(4-2) (2010) 1015-1032.

[10] D. E. Bell, L. J. La Padula, Secure computer systems: unified exposition and multics interpretation, MTR-2997 Rev. 1. Bedford: The MITRE Corporation (1976) 134.

[11] V. I. Gorodetsky, P. O. Skobelev, industrial applications of multi-agent technology: reality and perspectives, SPIIRAS Proceedings 6(55) (2017) 11-45. doi:10.15622/sp.55.1.

[12] I. A. Zikratov, T. V. Zikratova, I. S. Lebedev, A. V. Gurtov, Building a model of trust and reputation for the objects of multiagent robotic systems with decentralized control. Scientific and Technical Journal of Information Technologies 3(91) (2014) 30-38.

[13] I. A. Zikratov, I. I. Viksnin, T. V. Zikratova, Multiagent planning of intersection passage by autonomous vehicles. Scientific and Technical Journal of Information Technologies 16(5) (2016) 839-849. doi:10.17586/2226-1494-2016-16-5-839-849.

[14] E. I. Yurevich, I. A. Kalyaev, V. M. Lokhin, I. M. Makarov, Intellectual robots: Textbook for Universities, Moscow, Mashinostroenie Publ. (2007) 360.

[15] E. T. Jaynes, Probability theory: the logic of science, Cambridge: Cambridge University Press, (2007) 172.