1. Jakub POŚPIECH[1], 2. Witold NOCOŃ[1]

Silesian University of Technology, Gliwice(1)
ORCID: 1. 0000-0002-2425-4799; 2. 0000-0002-4785-958X

doi:10.15199/48.2024.08.46

Rules for efficient development of multiagent systems for continuous process control

***Abstract**. This paper presents a set of rules for a development of multiagent systems (MAS) for continuous process control. The purpose of presented guidelines is to provide a set of good practices that can be used to simplify development of the MAS, mostly during the design of its architecture and communication between agents. Prepared agent-based system based on provided rules is then verified in dissolved oxygen control for bioreactor aeration control system simulation.*

***Streszczenie.** Ta praca prezentuje zbiór zasad dla rozwijania systemów wieloagentowych (MAS) dedykowanych sterowaniu procesami ciągłymi. Zaprezentowane wskazówki mają na celu utworzenie zbioru dobrych praktyk, które mogę być wykorzystane do uproszczenia tworzenia MAS, przede wszystkim na etapach projektowania architektury systemu i sposobu komunikacji pomiędzy agentami. System agentowy przygotowany na podstawie zamieszczonych wskazówek jest następnie poddany weryfikacji podczas sterowania procesem napowietrzania bioreaktora. (**Zasady efektywnego rozwoju systemów wieloagentowych do ciągłej kontroli procesów**)*

Keywords: Agent-based control, control of dissolved oxygen concentration, switching control, multiagent system, simulational validation.
Słowa kluczowe: Aegntowe systemy sterowania, sterowanie stężeniem tlenu rozpuszczonego, systemy wieloagentowe, symulacyjna weryfikacja.

Introduction

Industrial solutions that incorporate multiagent systems (MAS) become gradually more popular. Nowadays, implementations based on multiagent approach can be found in many areas, ones that get the most research interest are cooperative, consensus, and MAS based formation control methods [1] that are applicable for purposes like UAVs navigation or distributed velocity control [2]. Active development of MAS can also be observed in areas of balance and fault control of microgrids [3] and mobile robot group control [4]. Over the years MAS were also introduced in many solutions dedicated to manufacturing systems control, these implementations are mostly used for balancing the workload of production machines [5] and creating flexible, reconfigurable manufacturing systems [6]. Applications of multiagent control system are also present in the process control area [7]. In [8] and [9] MAS is used to develop distributed controllers based on model-based predictive algorithms. Hierarchical, multi-controller incremental multiagent design framework for room temperature control is presented in [10]. MAS dedicated to pressure control of recycled gas with a presence of significant disturbances can be found in [11].

Although presented literature shows that research interest in developing MAS dedicated to continuous process control exists, these types of solutions in comparison with cooperative control or microgrid monitoring applications are adopted relatively slow. That situation is a result of many factors that obstruct development of agent-based solutions for continuous process control like concerns about time performance of most popular tools for building MAS [12]. Popular MAS design platforms cannot fulfill hard real-time constraints so they can be used only to control processes with high time tolerance like biotechnological processes [13]. Another important factor that slows down development of the MAS is scarce and outdated standardization [14, 15] and lack of common, easily accessible education on building agent-based systems [16]. In comparison with other popular programming paradigms, agent-based programming is characterized by a high level of abstraction. This makes agent-based systems better suited for solving complex problems but also makes development of such systems harder to learn. On the other hand, many MAS attributes are proven to be useful in process control domain e.g. reconfigurability [17], ability to perform social interactions [10, 18] or high degree of autonomy.

This paper presents set of guidelines for developing MAS for continuous process control. Introduced rules are focused on most important aspects of MAS development, system architecture and communication between agents. The purpose of this set of tips is to simplify the development of the MAS especially for developers unfamiliar with agent-based programming by providing generic advices regarding most important parts in MAS development. These rules are then verified by developing according to the rules a MAS dedicated to control a bioreactor aeration process.

The paper is organized as follows: in section 2 details regarding preparation of the rules are described. Sections 3 and 4 contain rules regarding development of system architecture and communication between agents respectively. Section 5 contains results of experimental verification of the rules. Lastly conclusions, discussion and authors' remarks are described in section 6.

Rules for MAS development

The purpose of the design rules is to present a set of good practices that can be used to simplify development of the MAS, mostly during the design of its architecture and communication between agents. Presented guidelines are generic and focus on the basics of MAS development. Lack of common education regarding MAS development and high abstraction level of agent-based programming makes starting development of MAS systems relatively difficult. Additionally, multiagent systems attributes make them best suited mostly for solving complex tasks which also raises difficulty of starting development in agent-based paradigm and makes the choice of using agent-based programming less attractive in comparison with approaches that are not as effective in solving complex task, but are simpler to use and already well known. Because of that, providing a comprehensive set of basic level rules of MAS implementation that are dedicated to developers unfamiliar with multiagent concept should be considered as an important part in popularization of multiagent based solution.

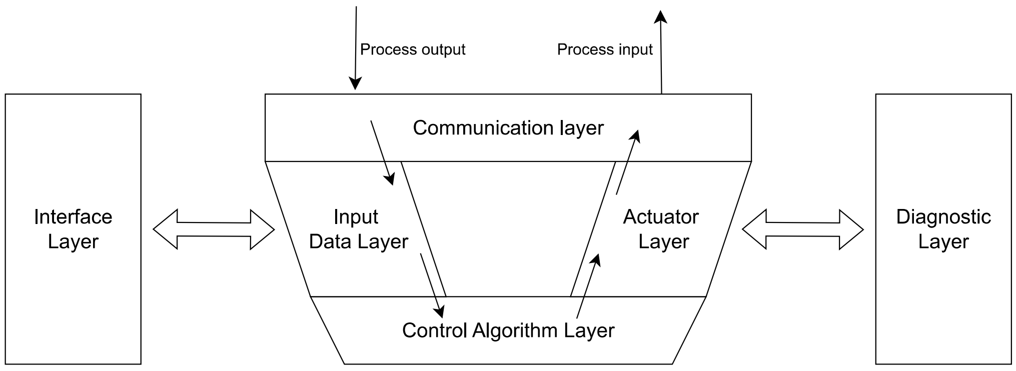

Fig. 1 MAS architecture layout.

MAS architecture design

Fig. 1 shows proposition of architecture layout for MAS dedicated for continuous process control. Arrows indicate information flow through the system which together with a controlled process forms a closed loop. Whole system is presented in the form of layers, each layer is responsible for one part of control task. Presented layout is generic and can be used for many approaches, not only agent-based.

In presented layout, layers in the center are responsible for controlling the process. Agents in Communication Layer send and receive data to and from controlled process. Information is then passed to the rest of MAS through agents in Input Data Layer which translate messages received from communication layer into the form used in the MAS. Control Algorithm Layer is dedicated for agents involved in calculations of control values. Derived control values are then passed to Actuator Layer that translates received logical control value (CV) into a form used by controlled process e.g. agents in this layer may create PWM signal from received CV. Translated signal is passed to agents in Communication Layer that send it to controlled process through specified protocol. Presented layout also includes Interface Layer and Diagnostic Layer. Agents in these layers are not directly dedicated to process control and can communicate with every other agent in the system to perform its tasks.

Agents are grouped in layers by their tasks, every layer should have defined messages types that can be received from and sent to other layers. Agents from one layer can communicate with agents from other layers using only these signals e.g. design might set that agents from Control Algorithm Layer can only receive process value (PV) and set point (SP) values from Input Data Layer and send CV to agents from Actuator Layer. Agents inside one layer can communicate with each other without any restrictions.

Each layer can be further divided in sublayers as in fig. 2 which shows exemplary division of the Input Data Layer. In presented case the layer is divided into two sublayers, agents in Sensors Sublayer are responsible to fetch data from the sensors and agents in System Input Sublayer provide other values that are not directly available from sensors like set point or simulation output that may be used by model-based controllers.

Dividing complex systems into smaller, well defined and clearly separated parts makes them easier to design, develop and maintain. Each layer forms almost independent subsystem thus in case of any modification in layer structure later integration of the changes would be in most cases limited to the scope of modified layer. Using layers for organizing MAS architecture has been already presented and has been verified for MAS architecture designs dedicated to other types of tasks like network [19, 20] and power flow management [21].

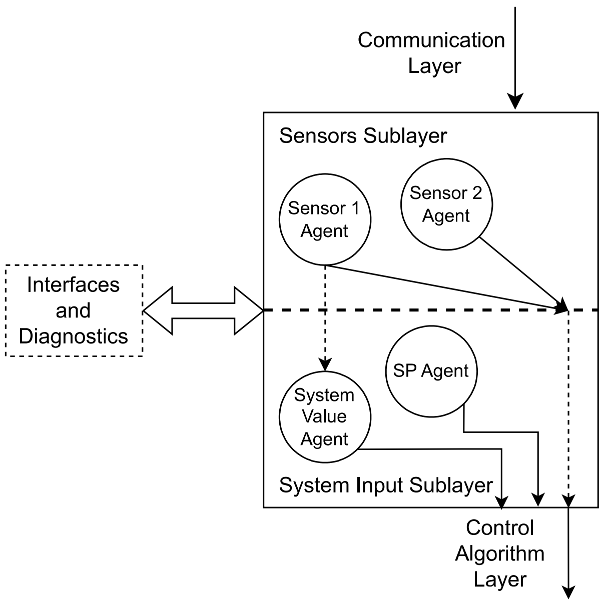

Fig. 2 Example of sublayers division.

Communication design

Proper communication between agents is necessary for efficient work of MAS, agents rely on communication between each other to cooperate, compete and perform other social acts in order to reach its goals. Agents are able to perform various communication acts, from simple passing of information to more complex behaviors like subscription mechanism or game theory problems.

Successful communication on any level of complexity requires, that agent understand content of each other's messages. Using ontology in information exchange ensures that agents have shared knowledge base and are able to understand their messages. Ontology can be adopted for MAS communication by using one of publicly available schemas that match the scope of work of prepared multiagent system. These knowledge bases are usually available in the form of large schemas with hundreds of classes, relations and instances. They are used mainly in large services to precisely describe concepts from broad area. Process control systems usually require concepts that are only a small parts of these knowledge domains i.e. knowledge about the process, sensors, actuators and environmental factors that may influence the process. Because of that, the scale of publicly available ontologies is in many cases too big for process control systems and could bring unnecessary complexity. In such cases it is more efficient to prepare dedicated lightweight ontology schema, fig. 3 presents a proposition of base layout of ontology schema for agent-based control systems for continuous processes.

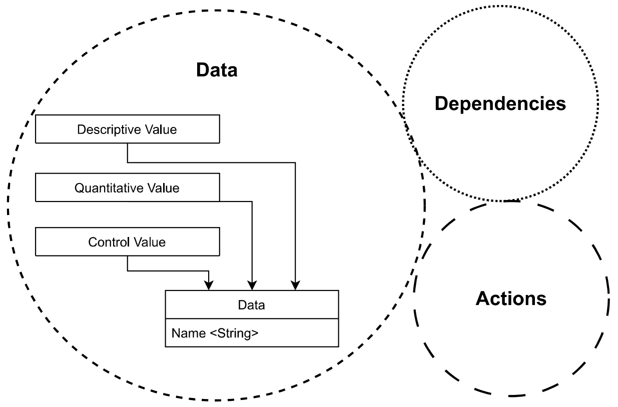

Fig. 3 Proposed ontology layout with base class hierarchy in data area.

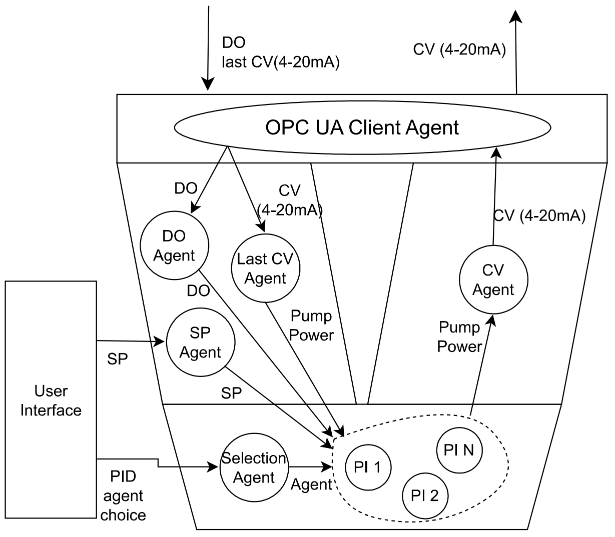

Fig. 4 MAS architecture for verification experiment

Proposed schema groups all concepts into three areas. Data area contains classes that represent all instances of data that agents exchange during process control task. Hierarchy inside the area shows example of base class organization inside the area, this hierarchy might be sufficient for systems using simple control algorithms like PID, but for more complex ones this schema would need to be extended according to the individual needs. Classes inside the Actions schema represent actions that can be taken by agents, these actions are often predefined in the environment used for agent-based development (e.g. FIPA Communicative Acts [22]). In such cases importing this set of actions directly into the ontology might be the most efficient approach. Diagnostic area contains classes that describe concepts that are not directly connected to process control but are used in MAS side tasks e.g. diagnostics or access control. Presented ontology areas division is not strict, classes from one area can use or inherit from classes in another area (e.g. concepts from *Actions* area will use concepts from other areas for detailed description of the actions), but in authors' experience building ontology according to these guidelines results in only few of such interactions.

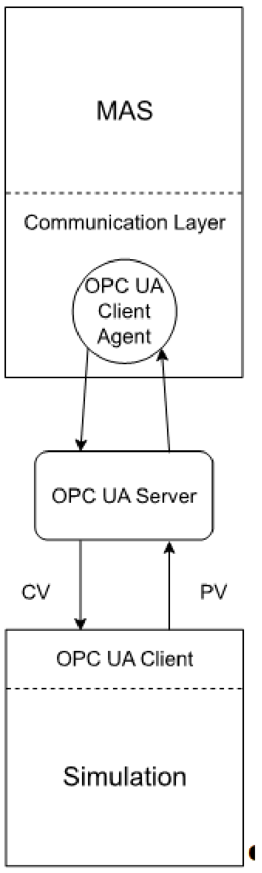

Fig. 5 Setup for verification experiment

Verification of rules

In order to verify presented rules MAS is built according to the guidelines and its functionality is tested by using the system to control simulated bioreactor aeration process. During the experiment MAS will use PI algorithms (each PI agent will have different tuning values) to tune aeration pump power according to the new SP of dissolved oxygen (DO) level. During process control user will be able at any time to manually select which PI agent should take over the control of the process, in that case, agents will switch control seamlessly.

MAS developed according to the rules is presented in fig. 4. Communication Layer contains one OPC UA Agent that exchanges data with simulated process through OPC UA protocol that connects MAS with simulation of the aeration process as presented in fig. 5. Input Data Layer is populated by three agents, all of them provide data to Control Algorithm Layer. Last CV Agent converts simulated 4-20 [mA] signal that controls aeration pump into 0-100% range that is used by controllers. SP Agent reads SP value from its user interface and DO Agent reads new DO values from plant messages and sends them further. Control Algorithm Layer contains Selection Agent that reads information about selected PI agent from its user interface and sends that information to all PI agents that currently reside in the system. PI agents realize PI control algorithm with specified tunings, PI agents may be added and removed from the system during runtime. At any time only one, selected PI agent calculates new CVs. Calculated CVs in the form of 0-100% are then passed to Actuator Layer where CV Agent converts it to 4-20 [mA] values and sends it to Communication Layer. Diagnostic Layer is omitted from the diagram because no diagnostic functionalities were implemented for verification purposes. In addition MAS uses ontology that is prepared based on layout from fig. 3. MAS was built using JADE that is Java framework for agent-based programming.

$$\frac{dD_O(t)}{dt} = k_L a(t)(D_{Osat} - D_O(t)) - OUR \quad (1)$$

Table 1 PI agents during experiment.

Agent name	Kc	Ti
PI 1	0.05	5
PI 2	0.01	12

Simulated aeration process is represented in (1), $k_La(t)$[1/h] is a transfer coefficient that represents a transfer of oxygen from air bubbles to liquid. Value of this coefficient is an input to the process because it is directly related to set power of aeration pump, it changes in the range of [0, 2.34] according to aeration pump speed. DO_{sat}[mg O_2/l] indicates DO saturation concentration, for simulation this value is set at 10[mg O_2/l]. OUR[mg O_2/lh] indicates oxygen uptake rate of the process, for the time of the experiments it is considered constant and equal to 11.88[mg O_2/lh].

Validation included three experiments that used two PI agents presented in table 1. Performed experiments included step change of SP value from 2[mg O_2/l] to 2.5[mg O_2/l]. During the first experiment control was performed exclusively by PI 1 agent, for the next experiment control was handled only by PI 2 agent, and for the last experiment at the beginning control was handled by PI 1 agent, but when DO value rose above 2.05[mg O_2/l] control was switched manually to PI 2 agent.

Result of the experiments are presented in fig. 6 and table 2. Control performed by aggressively tuned PI 1 agent resulted in lowest values of IAE and ISE indicators at the cost of DO oscillations. Results of control by passively tuned PI 2 agent show no oscillations and overshoot but presented control quality indicators have significantly higher values. Lastly results of experiment with switching control agents resulted in DO changes without oscillations, small overshoot and control quality indicators values close to the ones achieved during exclusive control of PI 1 agent.

Presented validation experiment and its results should be treated as a proof of concept of the proposed architecture design. Experiments like this can be performed using many other not agent-based approaches without any difficulty. In practice, MAS should be used in more advanced control scenarios with higher complexity of control algorithms, where attributes of the MAS like reconfigurability, autonomy of individual agents or ability to perform social interactions could be properly used.

Concluding remarks

This paper presented set of rules for development of MAS for continuous process control. Presented guidelines are generic and focused on the basics of MAS development. Such guidelines might be a solution to one of the issues that slow down the development of MAS in the continuous process control area – unfamiliarity of agent-based approach. Provided rules focus on two important parts of MAS design –systemic architecture and communication between agents. Guidelines for systemic architecture presented proposition of MAS layout where system is divided into layers, every system layer is then described and explained. Communication rules present set of tips to use during ontology preparation and proposed base of the layout that can be used during development of new, dedicated ontology.

Proposed set of rules was then verified in proof of concept MAS which was developed according to the guidelines and then tested during control of bioreactor aeration process simulation.

Later works on this topic should focus on further development and later standardization of the rules which would be helpful in popularization of agent-based control system.

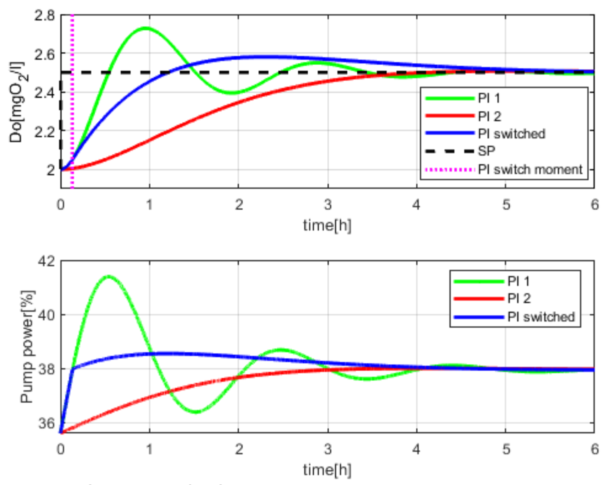

Fig. 6 Changes of DO and pump power during experiments.

Table 2 Control quality indicators for each experiment.

Experiment name	IAE	ISE
PI 1	1495	333.09
PI 2	2920	975.97
PI switched	1635	338.88

***Acknowledgments**: Authors were financed by the grant – subsidy for maintaining and developing the research potential in 2023, Jakub Pośpiech under grant 02/060/BKM23/0051 Witold Nocoń under grant 02/060/BK_23/0043.*

***Authors**: mgr inż. Jakub Pośpiech, dr hab. inż. Witold Nocoń prof. Pol. Śl., Silesian University of Technology, Department of Automatic Control and Robotics, 44-100, Gliwice, e-mail: jakub.pospiech@polsl.pl, witold.nocon@polsl.pl.*

REFERENCES

[1] L. Ding, Q. Han, X. Ge and X. Zhang, An Overview of Recent Advances in Event-Triggered Consensus of Multiagent Systems, *IEEE Transactions on Cybernetics*, 48 (2018), no. 4, 1110-1123, DOI: 10.1109/TCYB.2017.2771560.

[2] M. Sahal, T. Agustinah, A. Jazidie, and H. Du, Distributed Velocity Control in Cooperative Multi-Agent Moving Source Seeking, *Przegląd Elektrotechniczny*, 99 (2023), no. 8, 224–230, DOI: 10.15199/48.2023.08.40.

[3] P. Qaderi-Baban, M.B. Menhaj, M. Dosaranian-Moghadam and A. Fakharian, Intelligent multi-agent system for DC microgrid energy coordination control, *Bulletin of the Polish Academy of Sciences Technical Sciences*, 67 (2019), no. 4, 741-748, DOI: 10.24425/bpasts.2019.130183.

[4] R. Sikorski, Flexible multi-agent system for mobile robot group control, *Przegląd Elektrotechniczny*, 1 (2019), no. 12, 254–258,DOI: 10.15199/48.2019.12.57.

[5] S. Bussmann and K. Schild, An agent-based approach to the control of flexible production systems, in *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.01TH8597),* Antibes-Juan les Pins, France 2001, vol.2, pp. 481-488, DOI: 10.1109/ETFA.2001.997722.

[6] T. Arai, Y. Aiyama, Y. Maeda, M. Sugi, and J. Ota, Agile Assembly System by 'Plug and Produce,' *CIRP Annals*, 49 (2000), no. 1, 1–4, DOI: 10.1016/S0007-8506(07)62883-2.

[7] M. Metzger and G. Polakow, A Survey on Applications of Agent Technology in Industrial Process Control, *IEEE Transactions*

on Industrial Informatics, 7 (2011), no. 4, 570-581, DOI: 10.1109/TII.2011.2166781.

[8] M. Francisco, Y. Mezquita, S. Revollar, P. Vega and Juan F. De Paz, Multi-agent distributed model predictive control with fuzzy negotiation, *Expert Systems with Applications*, 129 (2019), pp. 68-83, DOI:10.1016/j.eswa.2019.03.056.

[9] J. Pospiech, Multi-Agent System for Closed Loop Model-Based Control of Dissolved Oxygen Concentration, in *2021 25th International Conference on Methods and Models in Automation and Robotics (MMAR)*, Międzyzdroje, Poland, 2021, 145-149, DOI: 10.1109/MMAR49549.2021.9528445.

[10] A. J. N van Breemen and T. J. A de Vries, Design and implementation of a room thermostat using an agent-based approach, *Control Engineering Practice*, 9 (2001), no. 3, 233-248, DOI:10.1016/S0967-0661(00)00111-8.

[11] Y. N. Guo, J. Cheng, D. Gong, and J. Zhang, A Novel Multi-agent Based Complex Process Control System and Its Application, in *Intelligent Control and Automation: International Conference on Intelligent Computing, ICIC 2006*, Kunming, China, 319–330. DOI: 10.1007/978-3-540-37256-1_39.

[12] G. Polaków, JADE environment performance evaluation for agent-based continuous process control algorithm, in *2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, Międzyzdroje, Poland, 2016, 571-576, DOI: 10.1109/MMAR.2016.7575199

[13] L. Ribeiro, S. Karnouskos, P. Leitão, J. Barbosa and M. Hochwallner, Performance Assessment Of The Integration Between Industrial Agents And Low-Level Automation Functions, in *2018 IEEE 16th International Conference on Industrial Informatics (INDIN)*, Porto, Portugal, 2018, 121-126, DOI: 10.1109/INDIN.2018.8471927

[14] S. Karnouskos, P. Leitao, L. Ribeiro and A. W. Colombo, Industrial Agents as a Key Enabler for Realizing Industrial Cyber-Physical Systems: Multiagent Systems Entering Industry 4.0, *IEEE Industrial Electronics Magazine*, 14 (2020), no. 3, 18-32, DOI: 10.1109/MIE.2019.2962225.

[15] S. Karnouskos and P. Leitão, Key Contributing Factors to the Acceptance of Agents in Industrial Environments, *IEEE Transactions on Industrial Informatics*, 13 (2017), no. 2, 696–703, DOI: 10.1109/TII.2016.2607148.

[16] V. Mařík and J. Lažanský, Industrial applications of agent technologies, *Control Engineering Practice*, 15 (2007), no. 11, 1364-1380, DOI:10.1016/j.conengprac.2006.10.001.

[17] G. Polaków, P. Laszczyk and M. Metzger, Agent-based approach to model-based dynamically reconfigurable control algorithm, in *2015 20th International Conference on Process Control (PC)*, Strbske Pleso, Slovakia, 2015, 375-380, DOI: 10.1109/PC.2015.7169992.

[18] D. Choiński, W. Nocoń and M. Metzger, Multi-Agent System for Hierarchical Control with Self-organising Database, in *Agent and Multi-Agent Systems: Technologies and Applications. KES-AMSTA 2007*, Wroclaw, Poland, 2007, 655-664. DOI:10.1007/978-3-540-72830-6_68.

[19] V. D. Chemalamarri, M. Abolhasan, and R. Braun, An agent-based approach to disintegrate and modularise Software Defined Networks controller, in *2022 IEEE 47th Conference on Local Computer Networks (LCN)*, Sep. 2022, 407–413. DOI: 10.1109/LCN53696.2022.9843585.

[20] S. T. Arzo, D. Scotece, R. Bassoli, F. Granelli, L. Foschini, and F. H. P. Fitzek, A New Agent-Based Intelligent Network Architecture, *IEEE Communications Standards Magazine*, 6 (2022), no. 4, 74–79, DOI: 10.1109/MCOMSTD.0001.2100053.

[21] M. Chen, S. D. J. McArthur, I. Kockar, and J. Pitt, Evaluating a MAS architecture for flexible distribution power flow management, in 2015 18th International Conference on Intelligent System Application to Power Systems (ISAP), Sep. 2015, 1–6. doi: 10.1109/ISAP.2015.7325531.

[22] FIPA Communicative Act Library Specification URL: http://www.fipa.org/specs/fipa00037/