Doctoral dissertation

Examining of feasibility and usefulness of agent-based control systems for controlling biotechnological processes.

Jakub Pośpiech

Supervisor: dr hab. inż. Witold Nocoń prof. Pol. Śl.

Auxiliary supervisor: dr hab. inż. Piotr Skupin prof. Pol. Śl.

Faculty of Automatic Control, Electronics and Computer Science

Doctoral School

Gliwice, 2025

Contents

Acknowledgements

The effort tied to preparation of a doctoral dissertation is not limited to its author, but is often shared with his closest family. Therefore, first of all I want to thank my wife, Justyna for her unlimited patience, support and involvement. I would also like to thank my parents and my parents in law for their care and comforting words whenever they were needed. Last but not least I thank my supervisors and colleagues from the department of Automatic Control and Robotics for their willingness to share their knowledge and experience, their help cannot be overstated.

I am grateful for having such great people around.

1. Introduction

Over the years, adoption of multi-agent systems (MAS) for industrial purposes drew attention of researchers focused on various types of processes. Constant flow of novel, more computationally powerful systems additionally increased accessibility and supported wider adaptation of MAS. On the other hand, some branches of industries, like biotechnological process control meet with relatively small count of research interest regarding MAS use, despite being considered as areas with high potential for introduction of such approaches. This dissertation is a summary of author's research focused on evaluating usefulness and feasibility of applications based on agent-based systems in biotechnological process control.

1.1. Agents and multi-agent systems

The use of term "agent" in the context of software development and artificial intelligence can be tracked back to 1970s [1, 2] when the first research works in this area has begun. The topic of "agents" started to focus increasing academic interest around 1990s. During following years many research facilities investigated the potential of agents and agent-based programming. These are also the times, when the discussion over a definition of what is an "agent" started to grow. The problem of vague agent terminology and lack of universal definition and standardization gained popularity [3]. In the multiple systematization attempts agents were described as e.g. "computer programs that simulate a human relationship by doing something that another person could do" [4], "persistent software entity dedicated to a specific purpose" [5]. Other researches tried to use more generic terminology like describing agents as "integrated reasoning processes" [6]. Difficulties in finding universal description of the agent were mostly caused by the fact that agents were investigated in the context of various areas like business management, telecommunication, modeling of social behaviors or manufacturing. Researchers from every field tried to use agents in a different way, according to their individual needs and as a result, opinions regarding the most important aspects of agents varied. These efforts were additionally complicated by the introduction of new terms based on "agents" like software agents [7], autonomous agents [8], mobile agents [9] etc. This situation caused some researchers to state that finding one definition that will match all use cases of agents is not possible [1, 10].

Currently, after around 30 years from the beginning of this discussion, there is still no standardized definition of an "agent" which might suggest that such universal term indeed could not be found. However, over the years some terminologies gained popularity and are widely used as agents' descriptions in research papers. The most popular is probably the concept of agent presented by Wooldridge and Jennings [11] which states that agents are autonomous beings with social abilities capable to act both proactively and reactively. Wooldridge later adapted this definition to prepare another, more generic statement that "an agent is a computer system that is situated in some environment and that is capable of autonomous action in this environment in order to meet its design objectives" [12]. Both definitions were used by the author of this dissertation to interpret what should be called an "agent".

Multi-agent systems are distributed systems that group multiple agents. The final result of MAS activity should be achieved by collective effort of all agents in the system, while each agent would try to reach its own designed objective. The main difference between MAS and a conventional distributed systems is an approach to design parts of the system. In case of MAS, every part of the system (i.e. agent) has set goal that it tries to autonomously achieve. There is no explicitly set global goal that all of agents are aware of and try to collectively achieve. Instead, if there is an objective set for the whole system, it should be achievable through agents reaching their individual goals. To reach specified objectives, agents might establish relations with other agents and use each other's capabilities through cooperation and coordination. However, agent's autonomy induces that these relations should not be strictly pre-defined, agents should start these relations by themselves. On the other hand, in conventional distributed systems, the global goal is often the only objective set and every part of the system is prepared with that goal in mind. Additionally, in these systems, relations between components are usually already predefined.

1.2. Biotechnological processes

Microorganisms are harnessed for many purposes in various branches of industry. Starting from a food industry where biological systems are used in e.g. bread, cheese and many other food and beverage production using traditional mechanisms known for centuries. More modern use cases include utilization of biotechnological systems in a large scale production of amino acids, antibiotics, enzymes, organic acids, pharmaceuticals or vitamins [13]. According to analyses, markets for products of biotechnological systems constantly grow [14]. Perspectives for future growth are also promising, mostly due to global focus on development of zero-waste and zero-emission industries, in which biotechnological systems are the main sources of biofuels, biogases and biopolymers. Wastewater treatment processes based on activated sludge also fit in this category [15]. Another potentially beneficial factor is the development of genetic engineering that enables better tailoring of microorganisms to specific processes by applying direct modifications to theirs genomes [16, 17]. Control of the biotechnological systems however is challenging - some signals cannot be measured, others are obtainable only through manual laboratory procedures making then unavailable online. On top of that, behavior of a microorganism colonies is often difficult to predict [18].

This dissertation focuses on application of MAS to two types of biotechnological processes used commonly in industry – activated sludge wastewater treatment, specifically control of dissolved oxygen (DO) concentration in this process and lactic acid fermentation.

Keeping a dissolved oxygen concentration at a desired level is one of main control tasks during the removal of nitrogen and phosphorus compounds in wastewater treatment. Dissolved oxygen concentrations outside of optimal range negatively impact process efficiency and in extreme cases may irreversibly degrade a microbial life inside a reactor. This process is considered difficult to control due to its nonlinearities and difficulties in measurements of activated sludge state and wastewater quality. Over the years many methods to control a dissolved oxygen concentration were proposed. Adaptive

backstepping algorithm is proposed in [19]. Other approaches to adaptive DO control include usage of coyote optimization algorithm [20] or reinforced learning [21]. In [22] authors proposed an approach where adaptation mechanism is implemented as an online procedure. Online procedures are also presented in [23], but in this case they are used to estimate oxygen uptake rate (OUR) used in predictive control algorithm. Other approaches, that incorporate predictive control algorithms are [24, 25] where data based, and genetic algorithms were used respectively. A Boundary-Based Predictive Controller (BBPC) is presented in [26]. In this case, predictive algorithm is used to control DO concentration using the ON-OFF pump. Dissolved oxygen control can be also considered as one of the objectives in multi-objective wastewater treatment control approaches [27, 28]. Other propositions include application of fuzzy neural networks [29], hierarchical control [30, 31], feedback-feedforward algorithms [32] or PWM control with Kalman filter used to estimate OUR [33].

Lactic acid (LA) is widely used as a substrate in chemical, cosmetics, food and pharmaceutical industries, it is also a base for poly-lactic acid – a biodegradable polymer [34]. The use of lactic acid constantly grows and is estimated to reach almost 2 million tons in 2025 [35]. Control of LA production is focused on maximizing the amount of produced lactic acid while minimizing the usage of substrates. Similarly to DO control, main difficulties come from high nonlinearities of the process and unavailability of online measurements of parameters directly related to microbial life inside the bioreactor [36]. Adaptive control algorithms for LA production control are presented in [37, 38], another approach to LA lactic acid production control uses Kalman filters as shown in [39, 40]. Genetic algorithms for LA control are presented in [41], in [42] authors present a sliding mode observer for estimation of biomass and lactic acid concentration based on glucose uptake. Lastly [36, 37] propose an approach where the control objective is oriented on substrate supply control.

1.3. MAS in industry

Over the years multi-agent systems were successfully applied for tasks from various areas of life and many branches of industry, providing a cogent proof of MAS versatility and ability to perform difficult tasks.

One of the most prolific fields for agent-based system application is a management and load control of smart grids. In this area, multi-agent systems are used for power flow management [43-48], energy market implementation [49, 50] and load shedding allocation [51]. Flexibility of agents and reconfigurability of agent-based systems is facilitated in self-healing smart-grid systems [52, 53].

Another area, where MAS ability to runtime reconfiguration is expected to be the most beneficial is manufacturing control systems [54, 55]. Reconfigurable MAS were implemented in Plug & Produce manufacturing systems [56, 57]. Besides implementation applied to factory test beds [58] these types of systems were also applied in large scale manufacturing plants for the automotive sector [59, 60], washing machine manufacturing [61] or mobile robots control [62].

Other industries, where MAS were successfully applied include transportation and logistics, where agent-based systems were used as traffic management systems in road

[63-66], rail [67, 68] and air [69-71]. Additionally, MAS was also used as routing and scheduling system [72,73]. Use cases of MAS adoption into systems related with oil production management can also be found in a literature [74]. Most of these implementations focus on oil plant and infrastructure supervision [75-77], but other applications like MAS based system for prevention of vandalism acts on oil infrastructure can be found as well [78]. MAS are also applied in modern smart-buildings, mostly for energy management, illumination and temperature control [79-83]. In a similar way to smart-grid management, MAS are also used in radio networks management to balance available resources load [84, 85]. An area where agent-based solutions were able to gain a significant popularity over last years is cooperative control and consensus algorithms [86]. These applications are used in formation control problem [87, 88] and to improve cooperation of multiple surveillance unmanned air vehicles (UAVs) [89].

For continuous processes, MAS is used in systems dedicated to decision support [90], supervisory [91], monitoring [92], fault detection [93] and as systems responsible for a direct control of the process. However, the last use case is met less often than the previous ones. Applications of MAS for process control can be found in gas pressure control [94], grinding process [95], incineration control [96], water level control [97], steel production [98] or internal combustion engine tuning [99]. Model-based reconfigurable control system is presented in [100]. However, according to author's best knowledge, MAS dedicated to biotechnological systems are scarcely found in the literature. Notable examples include multi-agent supervisory, knowledge-based systems and its application to the real wastewater treatment plant [101, 102], self-organizing MAS tested on a prey-predator problem simulation [103], agent-based decision support system for the wastewater treatment process [104] and MAS used to identify the state of the biotechnological process [105].

1.4. MAS potential to adaptation in biotechnological processes

Initial investigation of MAS potential to adaptation in biotechnological processes can be started by analyzing how well this type of processes fits in categories of problems that agent-based systems are considered efficient at solving. In [12] methodology for such analysis is presented, this approach consists of four statements, that should be evaluated in the context of the problem to solve. The statements are as follows:

a) "The environment is open, or at least highly dynamic, uncertain or complex"
b) "Agents are natural metaphor"
c) "Distribution of data, control or expertise"
d) "Legacy system."

According to author's analysis, statements a) and c) are good descriptions of the main challenges of biotechnological processes control. Complex dynamics and uncertainties resulting from measurement difficulties are one of the main obstacles in efficient biotechnological processes control. This type of processes is also characterized by high distribution of data – only part of the measurements is done automatically and available online. The rest is either unavailable (the only source of the data, if any, is through estimations) or available through manual, laboratory measurements. Remaining two factors might also describe biotechnological processes e.g. parts of wastewater

treatment (e.g. aeration or wastewater control loops) can be presented as a separate agent or group of agents. To minimize potential downtime, it is sometimes beneficial to not completely replace an old system with MAS but to integrate the legacy system with it instead. This analysis can be confirmed by [18], where authors present matching conclusions.

However, as presented in previous part of this chapter, the adoption of MAS into biotechnological processes is relatively poor, limited to the one large scale implementation of supervisory system over two decades ago and some preliminary, small scale research. In case of biotechnological processes, one of the most important features of control system candidates is their reliability and redundancy. In the biotechnological process control, similarly to other types of continuous processes, a potential downtime should be reduced to minimum. On the other hand, one of the main challenges regarding adaptation of multi-agent solutions is their lack of standardization and common knowledge about the development of agent-based systems. This slows down the introduction of these types of systems in large scale industrial applications [106-108]. These issues might additionally suggest that maintaining high reliability with agent-based systems in its current state might be significantly more difficult than in conventional approaches, where a design and verification procedures are strictly standardized.

In author's opinion in case of the biotechnological process control, ensuring MAS reliability should be considered at least as important as facilitating its advanced problems solving abilities. Based on multiple MAS use cases in various research this ability can be already considered well examined and proven. System's reliability should be ensured in both development and runtime phases. Developing fail proof systems usually requires an usage of established, well-known set of procedures and good practices. Runtime reliability can be improved by set of software features. In author's opinion one of the attributes of MAS that might positively impact its reliability is the ability to perform an online reconfiguration. This aspect of multi-agent systems was usually facilitated in manufacturing processes to reduce downtime between the change of production processes [58, 109]. However, it might also allow the MAS to better adapt to uncertain, changing conditions of controlled biotechnological process and to potential faults of control or measurement equipment which in turn might improve system's reliability and overall control quality by adjusting to changes inside the bioreactor.

As a result of this analysis the thesis is proposed that using an agent-based system, which is built based on a set of development rules for biotechnological processes, enables development of a control system with a reconfigurable structure which is able to adjust to conditions in controlled processes to achieve satisfactory control quality, improving MAS feasibility and proving its usability.

Both feasibility and usefulness are user oriented, subjective features. In the dissertation, usefulness is interpreted as evaluation whether the system is able to complete stated objectives, meets user's needs and provides or might provide in some specific circumstances an advantage over alternative solutions. The main criterion of usefulness evaluation stated in the thesis is the MAS ability to facilitate system's reconfigurable structure to optimize control quality.

Agent-based system feasibility will be assessed through feasibility studies. Feasibility studies are usually performed to evaluate project's potential to succeed by estimating probability of project's successful finish, its expected costs and projected benefits. Such studies are often performed by enterprises prior to start of large scale initiatives and are performed according to detailed checklists. However, in author's opinion this approach is not suitable for needs of the dissertation – projects are orders of magnitude smaller in scale and the feasibility is estimated in more general meaning. Because of that, author decided to use general approach to evaluate MAS feasibility i.e. answer two following questions:

- Are multi-agent systems capable of controlling biotechnology systems?
- Do benefits from MAS adaptation justify its use over alternative solutions?

To evaluate MAS feasibility author will try to answer these questions at the end of the dissertation using the evidence from its own studies and literature.

1.5. Structure of chapters

Chapter 2 presents available tools for multi-agent systems development, in chapter 3 processes used for verification of prepared MAS solutions are described. Chapters 4 and 5 are focused on pilot studies regarding implementation of MAS systems which led to preparation of set of development rules and the ontology presented in chapters 5 and 6. Chapters 7 and 8 describe feasibility studies on MAS implementations that apply prepared rules of development, lastly chapter 9 presents final conclusions.

2. Tools for MAS development

Growing popularity of agent-based solutions generated a demand for tools designed to simplify a development process of multi-agent systems. Various open and closed source frameworks for MAS development were published over the years. Some projects grew in popularity and are still developed today, others were replaced by more modern alternatives. This chapter provides short description of currently available agent-based frameworks, discusses current trends, explains author's MAS framework choice and describes technical limitations of frameworks in relation to control of real-time processes.

MAS based frameworks grew together with rising popularity of multi-agent systems. Development in this research field resulted in wide choice of tools for agent-based systems building. Ranging from general purpose open source platforms to proprietary, highly specialized toolkits. This chapter focuses on general purpose development frameworks.

JADE (Java Agent Development Framework) is one of the oldest but also the most popular agent-based frameworks currently in use. JADE is open source and uses Java to create multi-agent platforms [110]. It was first published in 1998 and is constantly developed up to this date [111]. According to many surveys it is the most popular MAS framework in both research and industry [112,113]. Authors of this tool declare full conformance with The Foundation for Intelligent Physical Agents (FIPA) agent-oriented specifications [114], which is source of widely accepted standards describing e.g. inter-agent communication. Over the years many additional tools were created that extend JADE functionality, e.g. WADE allows to build agent behaviors according to the workflow metaphor [115], BDI4JADE provides belief desire intention architecture on top of JADE [116]. Similar extension to BDI4JADE is JADEX that extends JADE with rational agents [117]. One of the newest extensions is JADESCRIPT created by the original authors of JADE as an answer to declared high entry level of agent-based programming. The purpose of JADESCRIPT is to simplify MAS development though introduction of beginner friendly scripting language [111, 118].

ASTRA might be considered as an alternative to JADE, which is also open-source and JAVA based framework. ASTRA is a combination and evolution of AgentSpeak and Teleo Reactive programming languages. It is designed in a way that should make it more accessible to developers with experience in lower-level languages like C [119].

JACK is also a Java based agent development framework, but accessible under proprietary license. JACK is an environment for building, running and integrating commercial-grade multi-agent systems based on belief desire intentions approach [120].

Historically Java was by far the most popular programming language used in MAS tools development [112]. However, according to author's recent observations, rise of agent-oriented tools based on Python is noticeable. One of the most established MAS frameworks based on Python is SPADE that over the years grew to a large scale comprehensive multi-agent systems platform. Its set of features includes e.g. IOT artifact connections or SPADE-LLM that allows integration of large language models (LLM) into MAS [121].

Other new, but quickly growing agent-based platforms built with Python are BSPL [122] and PIAF that similarly to JADE declares FIPA compliance [123].

Agent based frameworks grow and change dynamically over the years. In author's opinion, the best sources of up to date information regarding MAS frameworks are periodically published platforms reviews, one of current and the most comprehensive reviews can be found in [124].

Multi-agent systems in this dissertation are built using JADE framework. JADE was chosen because of its overall high popularity. JADE is used more often than other frameworks presented in this chapter. Such high popularity is resulting in a large number of tutorials and learning sources. This in fact is directly related with the topic of the dissertation. Large choice of learning sources indicates, that guidelines for developing JADE based agents are already prepared and thoroughly verified. Using these guidelines should simplify development process of single agents. This should allow author to pay more attention to the development of MAS in the context of biotechnological processes instead of struggling with technical difficulties regarding agents' implementation. Preparation of similar set of guidelines is one of the main objectives of the dissertation as stated in previous chapter. The additional positive factor is JADE's full FIPA compliance. Although standard's compliance is not strictly required in this research, it is in general a demanded feature in tools used in industry. This choice was successfully verified during the first MAS implementation described in chapter 4.

2.1. MAS tools in real-time environments

Control of real processes usually requires that control systems comply with hard real-time requirements. Presented MAS tools are built on Java and Python, which by design are not suitable in hard real-time environments what was also verified experimentally [125] (although these tools might be suitable for soft real-time tasks [126]).

Some approaches to overcome this limitation are proposed in literature. One of the approaches describes hybrid agent architecture, where agents are built from two components. Real time component is responsible of control and other related real-time aspects, while a non-real-time part is responsible to handle typical agent tasks [59, 127, 128]. In [126, 129] division of real-time tasks from MAS is proposed. In this hierarchical approach multi-agent systems runs in non-real-time regime, while all the hard real-time tasks are designated to a low level component. MAS can be either located at the same machine as the real-time component, it is then called an *on-device* system, or may reside in the other machine creating a *hybrid* system. Communication between agents and real-time components might be either direct, or performed indirectly e.g. through OPC UA server, these communication schemas are called *tightly coupled* and *loosely coupled* respectively.

In this dissertation experiments using real objects facilitate the *hybrid, loosely coupled approach*, while experiments based on simulations use the *on-device, loosely coupled* approach. In author's opinion setting up connection through well-established protocols like OPC UA is easier to maintain and more flexible than direct communication. *Hybrid* approach is used for real objects, because the ones used in this dissertation are open to academic society, making it difficult to set up and run a custom control systems locally, *hybrid* systems grant more flexibility in that regard.

3. Biotechnological processes used in MAS verification

Every agent-based control system presented in following chapters was verified by using it to control one of biotechnological processes. Whenever possible, systems were incorporated in control of a real test plant, otherwise process simulations were used. For every project in this dissertation, verification was performed on one of two processes - dissolved oxygen concentration control in wastewater treatment process or lactic fermentation control.

3.1. Dissolved oxygen concentration control

Keeping dissolved oxygen concentration at a desired level is an essential task for an effective control of many biotechnological processes, including wastewater treatment. Microorganisms living in a bioreactor require specific oxidizing conditions to maintain their growth. When DO concentration drifts from optimal levels, microorganisms might become unable to perform required actions at an expected rate. That can cause a significant drop of process efficiency [130]. In the worst case, the microbial life inside the bioreactor may deteriorate leading to irreversible changes to microorganisms composition and resulting in an expensive and time demanding restart of the process [131]. In case of the activated sludge wastewater treatment process, the dissolved oxygen concentration in an aeration phase should be kept around 2 (mg/l). Lower concentrations result in drop of nitrification efficiency and support faster growth of filamentous bacteria which, when in excess, cause difficulties in sedimentation stage. On the other hand, DO concentration level above 3 (mg/l) does not cause better treatment results and raises exploitation costs by raising energy consumption for air pumping[132].

In the dissertation, two versions of dissolved oxygen control in wastewater treatment process are used for verification of implemented MAS. The dissolved oxygen control simulation, implemented in LabVIEW environment is mostly used for preliminary verification, while final measurements are performed on the real, laboratory scale test plant.

3.1.1. Laboratory scale test plant

Schema of laboratory scale test plant used in MAS verification is presented in fig. 3.1 and real plant is presented in fig. 3.2. The laboratory setup includes a well-mixed fed-batch bioreactor with constant liquid level, where biological removal of organic waste is performed. DO is measured with the sampling time 1 (s) using Hach Lange SC200 meter with a dissolved oxygen probe. Biomass inside the bioreactor is fed with substrate transported by a peristaltic pump, substrate flow can be manipulated to induce changes of process load. During experiments organic substrate prepared from 6 (g) of peptone per 1000 (ml) of water is fed to the reactor by the 150 (ml/h) peristaltic pump. Sensors and most of the actuators are connected to the NI-9074 CompactRIO controller, a variable speed pump is connected to the separate NI-9074 CompactRIO controller. The ON-OFF air pump is limited to run in 30 (s) aeration cycles. During each aeration cycle it is turned ON for 2 (s). ON-OFF pump’s air dosage needed to be limited because it has too high efficiency in relation to reactor needs. The last part of the setup is a desktop PC that

exchanges data with controllers using LabVIEW Network-Published Variables and publishes variables using OPC UA server created with LabVIEW OPC UA Toolkit. [133]

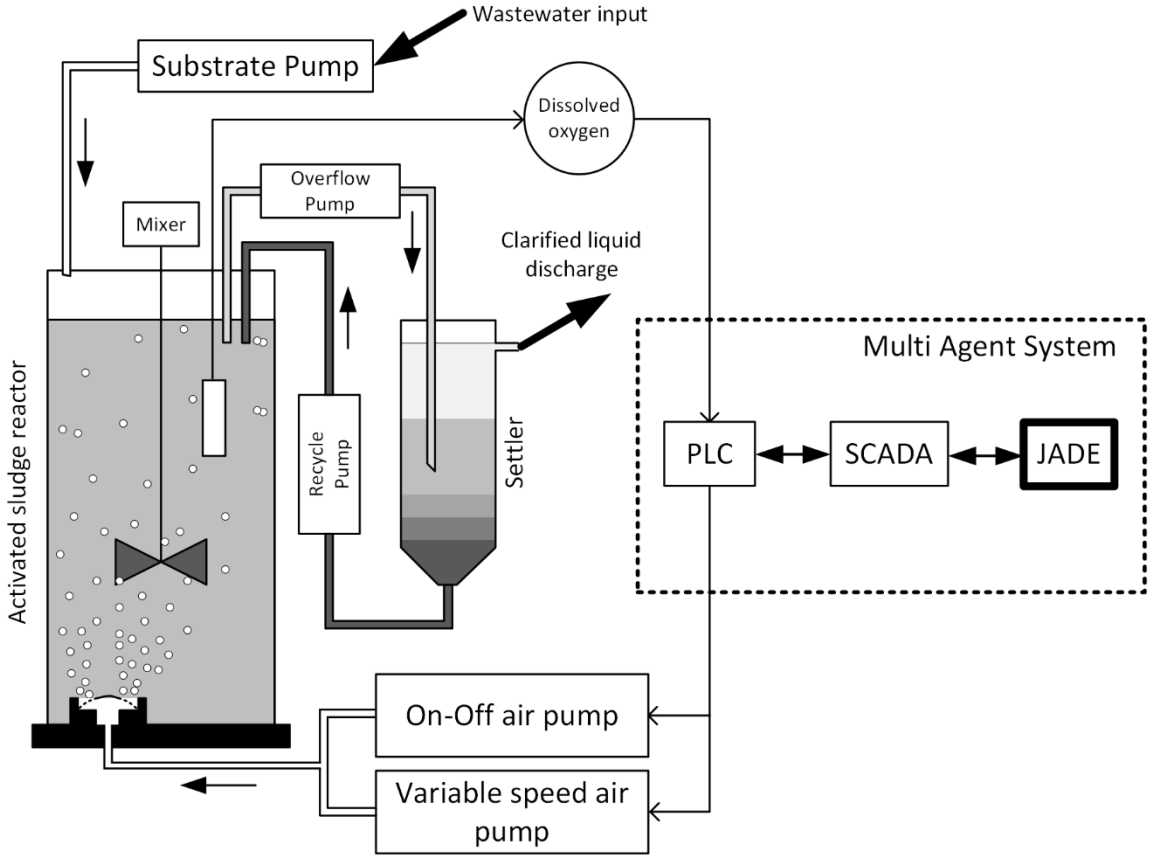

Fig. 3.1 Test plant layout

Fig. 3.2 Laboratory scale wastewater treatment test plant used in MAS verification

3.1.2. Dissolved oxygen concentration model

Agent-based systems presented in this dissertation often use controlled process's model for a control value calculation. DO concentration dynamical changes model used for this purpose is presented in (3.1).

$$\frac{dDO(t)}{dt} = k_L a(t)\big(DO_{sat} - DO(t)\big) - OUR(t) \quad (3.1a)$$

$$T_{k_L a}\frac{dk_L a(t)}{dt} = -k_L a(t) + u_{air}(t - T_0) \quad (3.1b)$$

In this model $k_{La}(t)$ (1/h) represents the transfer coefficient of oxygen from air bubbles to liquid, DO_{in} is DO concentration of input flow (mgO_2/l). DO_{sat} is DO saturation concentration (mgO_2/l). DO_{sat} is considered constant because all the experiments are performed using indoor reactor where external conditions, especially temperature can be considered constant at 20°C. However, in practice DO saturation concentration can change depending on e.g. pressure, air humidity and primarily water temperature [134]. Oxygen demand of organisms inside the reactor is represented by $OUR(t)$ ($mgO_2/l\ h$), this value depends on substrate flow and its changes are the main source of process disturbances. First-order plus dead time model in (3.1b) contains the following parameters: time constant T_{kLa} (h), delay time T_0 (h) and the control action represented by the term u_{air}(t). For the ON-OFF pump value of this term is switched between 0 (OFF mode) and u_A (ON mode). For aeration with the variable speed pump values of this term depend on parameters of the running pump. Process is operated with constant liquid level and under unvarying mixing conditions so values of parameters T_{kLa} and T_0 are considered constant [133].

This model is based on author's previous works presented in [26] and is a modified version of widely used aeration model [135-137] presented in eq. (3.2). The first factor was removed from the final model because DO concentration of input flow DO_{in} (mgO_2/l) is unknown and in presented test plant ratio between input flow $Q(t)$ which is constant c.a. 0.6 (ml/s) and reactor's volume V (15.6 (l)) is low enough, that the factor might be omitted without significant loss of accuracy. This simplification can be often met in the literature [137-140].

$$\frac{dDO(t)}{dt} = \frac{Q(t)}{V}(DO_{in}(t) - DO(t)) + k_L a(u_{air}(t))\big(DO_{sat} - DO(t)\big) - OUR(t) \quad (3.2)$$

Addition of equation that modifies $k_{La}(t)$ characteristics from control-based to control and time based is caused by observations of DO changes during normal work of laboratory test plant. Examples of such changes are presented in fig. 3.3. Model (3.2) assumes that k_{La} responds instantly to control modification. Assuming that OUR changes are significantly slower, changes in control should result in "sharp" changes of DO trajectory. As can be seen in fig. 3.3, this assumption does not match the real measurement data gathered from laboratory test plant. Change of DO trajectory is stretched over some period of time after control modification. However, this do not indicate that model (3.2) is incorrect. In case of presented test plant, described phenomenon is caused mostly by internal dynamics of dissolved oxygen probe and in the smaller scale by an internal characteristics of an air pump and dynamic behavior of air bubbles dispersion over liquid. As a result, eq. (3.1b) does not model real dynamical changes of k_{La} that in reality behaves like in (3.2), but models internal dynamics of DO measurements that are not explicitly

included in the model. Addition of eq. (3.1b) caused, that modelled DO changes could more closely follow real changes, which was proven in [26].

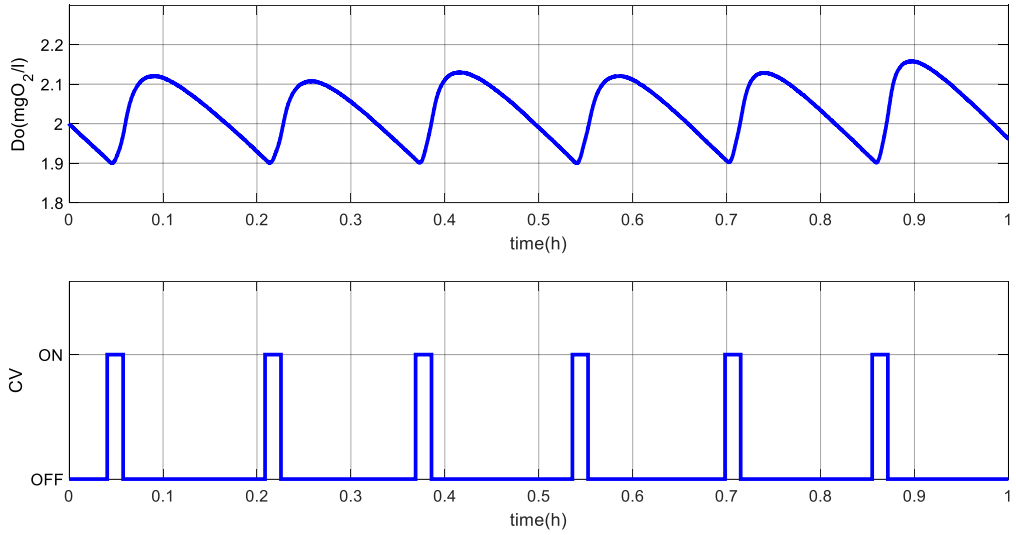

Fig. 3.3 Typical DO changes in laboratory test plant

3.1.3. DO model identification procedure

Some of model-based multi-agent controllers presented in following chapters require initial off-line identification of values of DO concentration model parameters presented in previous chapter. This identification is performed using the procedure presented in [26], that uses a single aeration cycle with an ON-OFF pump like presented in fig. 3.4. This process can be divided into 3 stages performed in order:

- Identification of *OUR* value – firstly, *OUR* value is estimated from a decreasing slope od DO concentration changes. *OUR* changes are slow enough that it is safe to assume that for single aeration cycle *OUR* is constant. Two points are selected - t_A occurs 4 minutes after the detected peak value of DO concentration, t_B is the moment, when aeration is turned ON again. For this time period DO falls in relatively constant rate meaning that it is safe to assume that k_{La} is constant and equal to 0. Using this assumption, eq. (3.1a) can be reduced to $\frac{dDO(t)}{dt} = -OUR$. Then, *OUR* can be directly derived from the DO slope as in (3.3).

$$OUR = \frac{DO(t_B) - DO(t_A)}{t_B - t_A} \tag{3.3}$$

- Computation of $k_{La}(t)$ – using calculated *OUR* it is possible to derive $k_{La}(t)$ values in discrete time moments, by rearranging eq. (3.1a) as presented in (3.4). Indices *i* and *i-1* correspond to current and previous discrete moments of time and Δt is a sampling time. This stage includes numerical differentiations that are known to be sensitive to measurement disturbances. Because of that, it is often necessary to filter DO signal with e.g. a moving average filter to cancel noise influence.

$$k_L a_i = \frac{DO_i - DO_{i-1}}{\Delta t (DO_{sat} - DO_i)} + \frac{OUR}{DO_{sat} - DO_i} \tag{3.4}$$

- Estimation of u_A, T_{kLa} and T_0 – constructed $k_{La}(t)$ is used to derive remaining 3 parameters. After manual identification of the most suitable T_0, u_A and T_{kLa} can be calculated using a least-squares method.

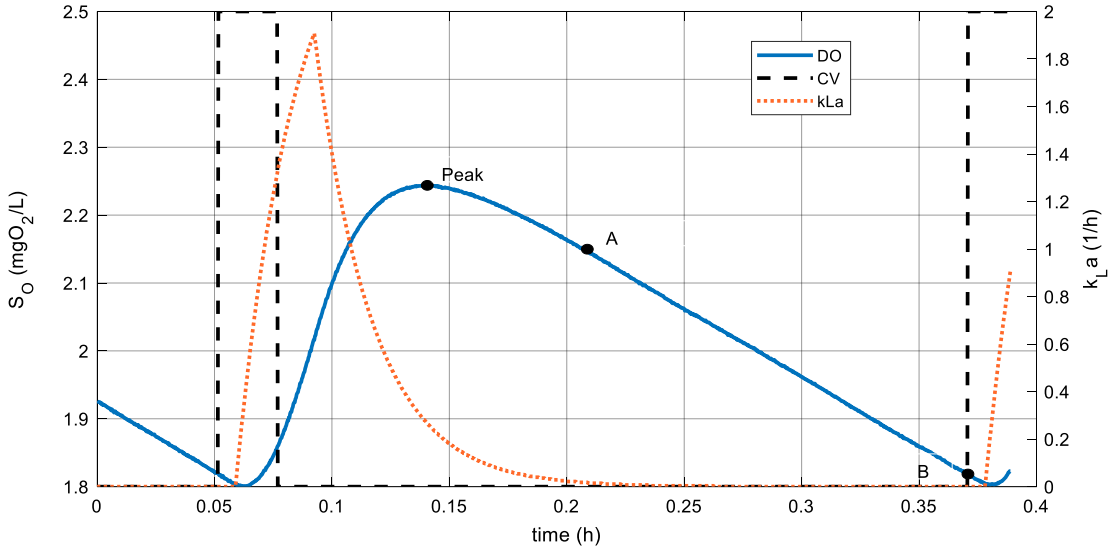

Fig. 3.4 Illustration of model identification parameters

3.1.4. Simulation of dissolved oxygen concentration control

Simulation is built in LabVIEW environment and consists of two modules presented in fig. 3.5.

OPC UA Server module implements a server that handles communication between the simulation, the control system (that in this dissertation will be always implemented using agent-based approach) and the actual simulation.

Simulation uses the model (3.1) to simulate behavior of dissolved oxygen concentration in response to control changes. The front panel view of the simulation is presented in fig. 3.6.

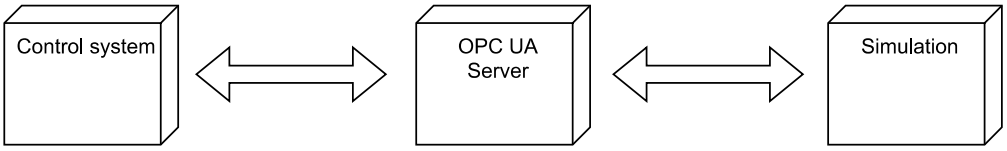

Fig. 3.5 Simulation modules

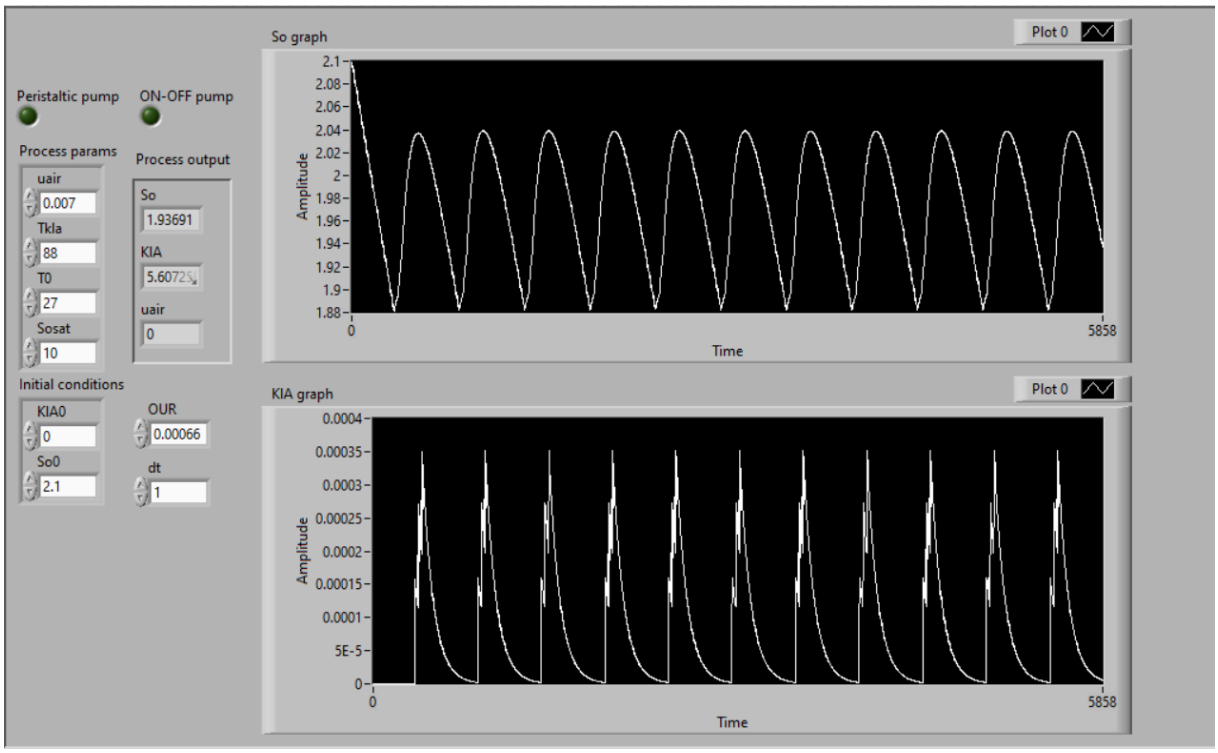

Fig. 3.6 Simulation front panel

3.2. Lactic fermentation control

Fermentation processes are one of the most widespread processes in a biotechnological industry. Lactic fermentation process is focused on obtaining lactic acid, usually from glucose [36, 37]. Production of lactic acid is used by many branches of industry, including pharmaceutical industry, cosmetics, chemicals, food and polymer industry, where production and usage of lactic acid significantly rises over the years. Lactic acid is a base for polylactic acid (PLA) production – eco-friendly, non-toxic polymer that is expected to replace an oil based polymers in many cases [34, 141]. Control goal in this process is to maximize lactic acid production and minimize residual glucose concentration in process output. Too high residual glucose concentration reduces cost efficiency of the process and in some cases might spoil product of the reaction [36]. Lactic acid fermentation control provides similar challenges to other biotechnological, especially fermentation processes. These challenges include high nonlinearities, presence of parameters with highly uncertain, time varying values and difficulties with automatic measurements of process parameters due to lack of, or high cost of measurement sensors. Agent-based control systems presented in this dissertation are verified using a simulation of a two, constant volume reactor tank process [36, 37]. Layout of the system is presented in fig. 3.7. Eq. (3.5-3.19) present lactic fermentation model implemented in the simulation.

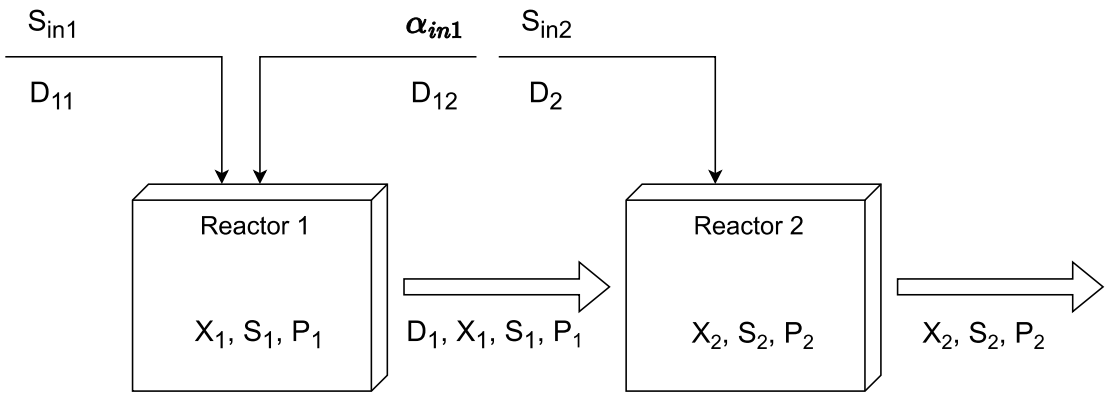

Fig. 3.7 Lactic fermentation reactor setup

$$\frac{dX_1}{dt} = (\mu_1 - k_d)X_1 - D_1X_1 \quad (3.5)$$

$$\frac{dP_1}{dt} = \nu_{p1}X_1 - D_1P_1 \quad (3.6)$$

$$\frac{dS_1}{dt} = -q_{s1}X_1 + D_{11}S_{in1} - D_1S_1 \quad (3.7)$$

$$\frac{d\alpha_1}{dt} = D_{12}\alpha_{in1} - D_1\alpha_1 \quad (3.8)$$

$$D_1 = D_{11} + D_{12} \quad (3.9)$$

$$\frac{dX_2}{dt} = (\mu_2 - k_d)X_2 + D_1X_1 - (D_1 + D_2)X_2 \quad (3.10)$$

$$\frac{dP_2}{dt} = \nu_{p2}X_2 + D_1P_1 - (D_1 + D_2)P_2 \quad (3.11)$$

$$\frac{dS_2}{dt} = -q_{s2}X_2 + D_1S_1 + D_2S_{in2} - (D_1 + D_2)S_2 \quad (3.12)$$

$$\frac{d\alpha_2}{dt} = D_1\alpha_1 - (D_1 + D_2)\alpha_2 \quad (3.13)$$

$$\mu_i = \bar{\mu}_{max\,i}\frac{\bar{K}_{P_i}^{gc}}{\bar{K}_{P_i}^{gc}+P_i}\frac{S_i}{\bar{K}_S^{gc}+S_i}\left(1 - \frac{P_i}{P_C^{gc}}\right) \quad (3.14)$$

$$\nu_{pi} = \eta\mu_i + \beta\frac{S_i}{\bar{K}_{S_i}^{rc}+S_i} \quad (3.15)$$

$$q_{si} = \frac{\nu_{pi}}{Y_{PS}} \quad (3.16)$$

$$\bar{\mu}_{max\,i} = \frac{\mu_{max}(\alpha_i - \alpha_0)}{K_{\alpha\mu}+(\alpha_i - \alpha_0)} \quad (3.17)$$

$$\bar{K}_{P_i}^{gc} = \frac{K_{Pmax}^{gc}(\alpha_i - \alpha_0)}{K_{\alpha P}+(\alpha_i - \alpha_0)} \quad (3.18)$$

$$\bar{K}_{S_i}^{rc} = \frac{K_{Smax}^{rc}(\alpha_i - \alpha_0)}{K_{\alpha S}+(\alpha_i - \alpha_0)} \quad (3.19)$$

In the presented model X_i, S_i, P_i and α_i refer to concentrations of biomass, substrate, lactic acid and enrichment factor in i-th reactor. μ_i is a biomass growth rate, υ_{Pi} is specific rate of lactic acid production and q_{Si} is specific rate of glucose consumption. D_{11} and D_2 are dilution rates of influent glucose concentrations S_{in1} and S_{in2} respectively, D_{12} is the dilution rate of the influent enrichment factor α_{in1}. $\bar{\mu}_{max\,i}$ is the maximum specific growth rate $\bar{K}_{P_i}^{gc}$ is the lactic acid inhibition constant, $\bar{K}_S^{gc}$ is the affinity constant of growing cells, P_C^{gc} is the critical lactic acid concentration, $\bar{K}_{S_i}^{rc}$ is the affinity constant of resting cells for glucose and Y_{PS} is the constant substrate to product conversion yield. η and β are positive

constants. Superscript gc describes parameters related to growing biomass, while rc relates to resting biomass. α_0 is the minimal nutritional growth factor, $K_{\alpha\mu}$, $K_{\alpha P,}$ $K_{\alpha S}$ are saturation constants, μ_{max}, $K_{P_{max}}^{gc}$ and $K_{S_{max}}^{rc}$ are limit values for each parameter.

Similarly to dissolved oxygen model simulation in this case LabVIEW was also used as a simulation environment and simulation itself can be divided into two modules, like in fig. 3.5.

4. Multi-Agent System for dissolved oxygen concentration control

This initial research was focused on preparation of MAS that would be able to control dissolved oxygen concentration during removal of organic waste in a well-mixed bioreactor inside a laboratory environment using ON-OFF control. Dissolved oxygen control is described in detail in the chapter 3, the laboratory setup used during that research is presented in fig. 3.1. The main objective during this study was to prepare a MAS with a modular structure allowing modification, replacement and removal of residing agents with minimal influence on the rest of the system. MAS shall incorporate control algorithms that would vary in complexity and reliability. Control system would need to be able to deal with failures of any control algorithm by seamlessly switching between implemented controllers when more accurate control is available. DO concentration should be close to 2 (mgO_2/l), optimally inside the range of 1.95-2.05 (mgO_2/l). Results of this research were presented at the 25th International Conference on Methods and Models in Automation and Robotics (MMAR), this chapter is based on paper presented during this conference [131].

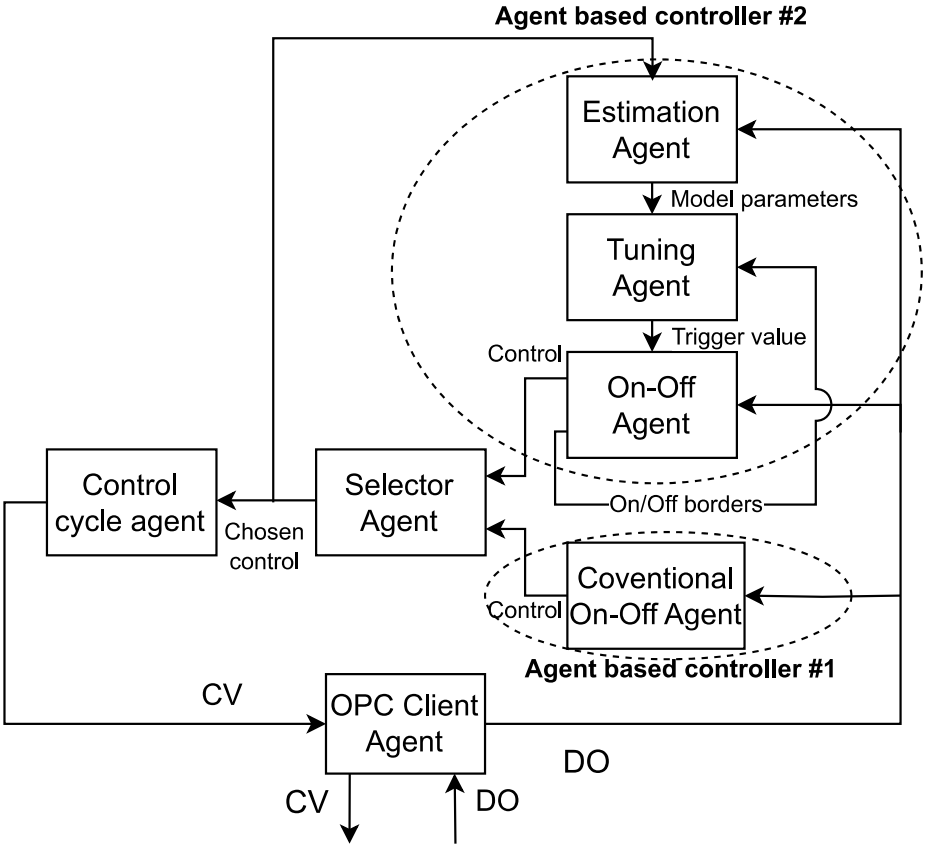

Fig. 4.1 Schematic diagram of designed MAS for DO concentration control

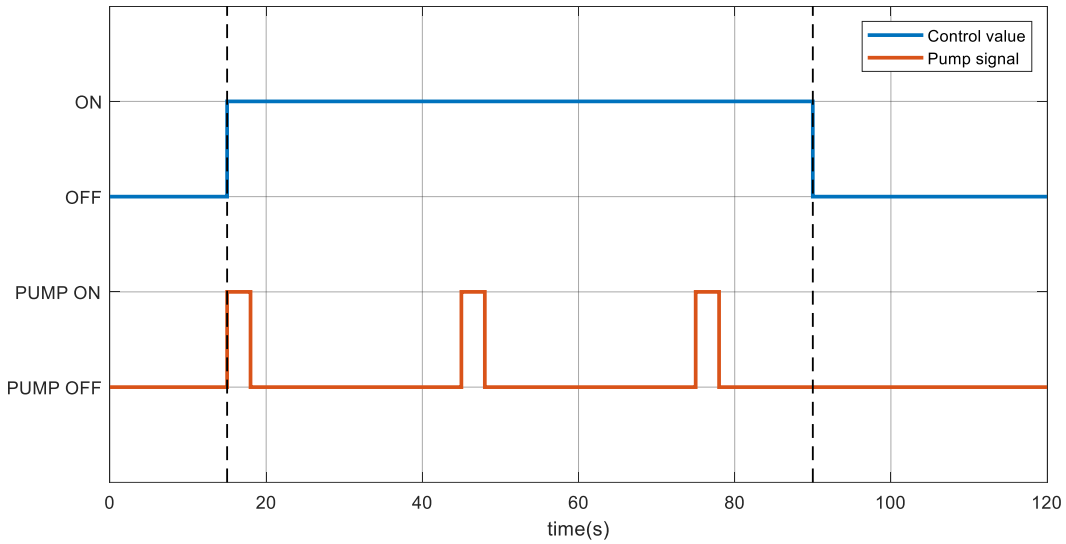

Fig. 4.2 Transformation of control value into pump control signal

4.1. MAS Architecture

Prepared architecture, presented in fig. 4.1. Agents inside this schema are organized in a loop with the OPC Client Agent functioning as a gateway for system's input and output data. Every agent or indicated group of agents is considered as a separate module that can be modified, replaced or in case of Agent Based Controller groups even removed without need to alter logic or procedures inside other modules in the system.

OPC Client Agent is an interface between external network and the MAS. During every control cycle OPC Client Agent receives data from controlled process and sends control values to the actuators using OPC UA communication protocol. OPC UA interface is configured using an open source stack called Eclipse Milo.

Control Cycle Agent is responsible for adapting control values prepared by the MAS and received from Selection Agent into form used by a physical actuator. In this case it transforms ON control value into series of 30 (s) cycles during which the aeration pump is turned on for 2 (s). This transformation is visualized in fig. 4.2. It is the only agent in the system aware of real actuator's characteristics, therefore only this agent would require modification in case of changes to the actuator. Transformed signal is sent to OPC Client Agent.

Selector Agent collects all control values prepared and sent by Agent Based Controllers and selects the best one. In every control cycle Selector Agent searches for controllers in a DF (Directory Facilitator) catalogue [142] and sends a call for control value proposal to every controller found. After the call, Selection Agent waits specified amount of time for controllers responses and then begins a selection procedure. In this case, the selection is based on controllers' priority arbitrarily assigned by an user. Selector Agent selects control value provided by a controller with the highest priority and propagates it to agents that earlier reported their need to receive selected control value (i.e. Control Cycle Agent and Estimation Agent). If the highest priority controller fails to deliver a control value, Selector Agent selects control provided by an agent with second highest priority etc.

In the created MAS #1 controller serves as a backup controller and has arbitrary priority assigned to 0, while preferred #2 controller has priority assigned to 1.

Agent Based Controllers are modules built from one or more agents designated to determine the control value based on provided information about process values and previous control values. Agent Based Controllers calculate new control values propositions in response to incoming new data from controlled process. Control propositions are sent to Selection Agent when a call for proposals is received. Multiple controllers would compete with each other to prepare the control value that would provide the best control quality. Besides competition, these modules also cooperate with each other – if a preferred controller fails to deliver the control value, other controllers will step up to provide seamless switch between control algorithms and guarantee a control continuity. The MAS includes two controllers, the controller denoted as #1 in fig. 4.1 implements conventional ON-OFF control while #2 uses a model based ON-OFF control that is described in detail in the next chapter.

Agents gain information about presence of other agents in the system through agent descriptions in the FIPA specified DF catalogue [142] and exchange messages complaint to "FIPA ACL Message Structure Specification" [143] which is a part of JADE framework. In cases where passing additional data through message content is needed, it is provided as "*parameter:value"* couples separated by a semicolon.

4.2. Multi-agent model based ON-OFF controller

Created multi-agent ON-OFF controller is composed of three cooperating agents. Estimation Agent calculates parameters' values of reduced version of the model (3.1) presented in (4.1a) and (4.1b) based on knowledge of output and input values of the process. $y(mgO_2/(l*s))$ corresponds to a dynamical traits of aeration process. Aeration processes have usually non-linear characteristics. However, because of the control goal focused on reaching single operating point, linear simplification of process's dynamics could in this case provide an acceptable approximation. $T_0(s)$ denotes time delay of the process, $T_y(s)$ is interpreted as time constant in (4.1b) and $u_y(mgO_2/(l*s))$ value switches between: u_{on} when pump is switched ON and 0 when it is turned OFF.

$$\frac{dDO(t)}{dt} = y(t) - OUR(t) \tag{4.1a}$$

$$T_y \frac{dy(t)}{dt} = -y(t) + u_y(t - T_0) \tag{4.1b}$$

$OUR(mgO_2/(l*s))$ is calculated from a slope of declining DO concentration value. When control is in low state. OUR is estimated once every 30(s) with the least-squares method using measurement data from last 30(s). Estimation is saved as new OUR value only if difference between this and the last two calculated values is less than 5% of the new OUR. Otherwise new value is discarded and last OUR that fulfilled this requirement is still in use. OUR changes over time are slow enough so it can be assumed that during single 30(s) estimation period OUR is constant. Using this assumption with the model (4.1), the difference between starting and ending DO concentrations for OUR estimation can be described with eq. (4.2a), where DO_0 and y_0, are values of DO and y at the start of the estimation and τ is sampling period. Equation (4.2) shows that difference in DO

concentrations depends on two factors, OUR and an expression associated with aeration process's dynamic traits, which decreases over time. The restriction of three consecutive estimation values differing by less than 5% is dedicated to ensure that the influence of *y(t)* to OUR calculations is marginal, thus eq. (4.2a) can be reduced to (4.3) enabling use of the least squares method to estimate OUR. Magenta dots on the declining DO slope in fig. 4.3 indicate moments in time when calculation of new OUR was performed.

$$DO_0 - DO(\tau) = OUR \cdot \tau - T_y(y_0 - y(\tau)) \tag{4.2a}$$

$$y(\tau) = y_0 e^{-\frac{\tau}{T_y}} \tag{4.2b}$$

$$DO(\tau) = DO_0 - OUR \cdot \tau \tag{4.3}$$

Estimation of remaining parameters is based on the data gathered between the beginning of aeration and the moment when DO concentration starts to rise. Delay time T_0 is identified using statistical analysis of DO concentration variations by measuring elapsed time between the start of aeration and the moment when differences between subsequent DO values starts to significantly diverge (differ by more than a standard error multiplied by a chosen constant) from their mean values. Graphical interpretation of this parameter is presented in fig. 4.3. After identification of T_0 Estimation Agent calculates time constant T_y, and u_{on} with the least-squares method. Newly identified parameters are then compared with the previous values by analyzing how accurately both models can simulate the last aeration cycle. Agent compares values of integral square errors (ISE) and absolute differences between their minimal values which graphical presentation is shown in fig. 4.4. New parameter values are accepted only when simulation with these parameters has lower values for both criteria. After successful identification Estimation Agent sends new parameter values to Tuning Agent.

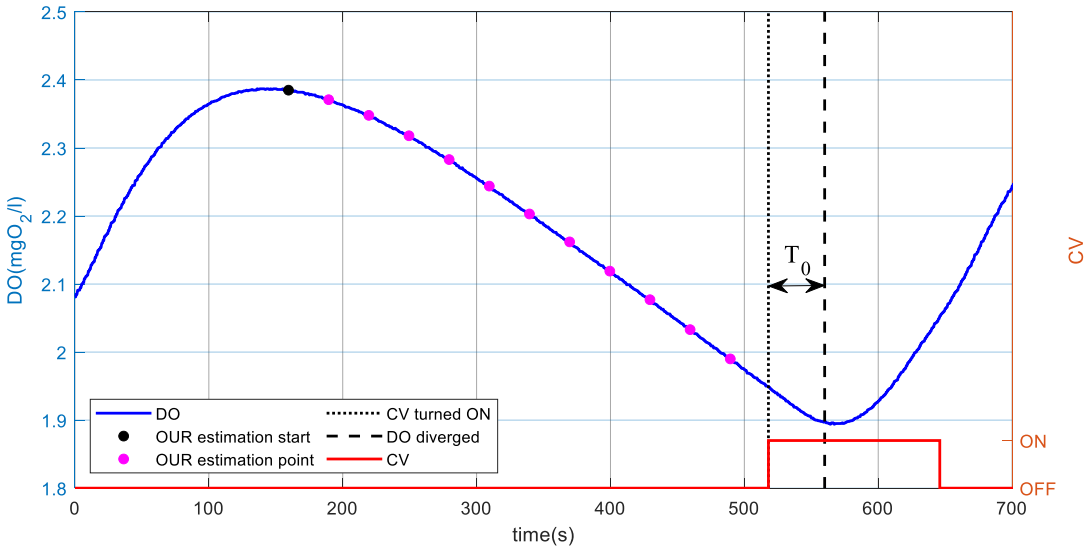

Fig. 4.3 Representation of *OUR* and T_0 estimations

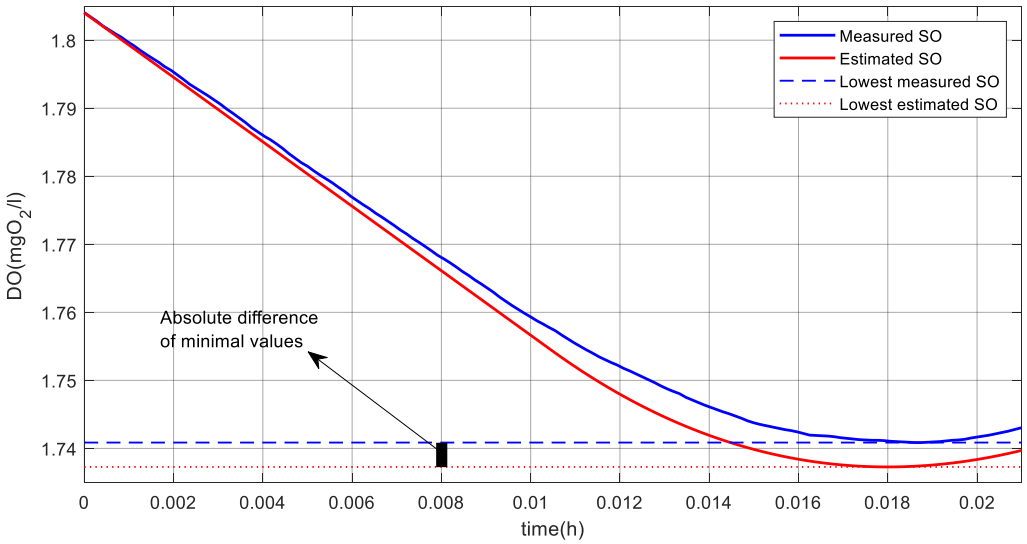

Fig. 4.4 Graphical presentation of absolute differences between minimal values

Tuning Agent uses model (3.1), parameters provided by Estimating Agent and boundary values SP_{min} and SP_{max} to find optimal time for switching on the pump to precisely reach SP_{min} value. This value is denoted as $SP_{trigger}$ and its graphical presentation is shown in fig. 4.5. $SP_{trigger}$ is a maximal SP_i value from range $<SP_{min}, SP_{max}>$ for which the lowest predicted concentration of DO in a single aeration cycle $\widehat{DO}_{mini}$, is lower than SP_{min}. Tuning Agent looks for $SP_{trigger}$ by performing iterative predictions of *DO* behavior, assuming that pump will be switched on when oxygen concentration reaches SP_i. Identification starts from $SP_0=SP_{max}$, each consecutive SP_i is calculated using formula $SP_i=SP_{i-1}-0.01$, this process stops when either $\widehat{DO}_{mini} < SP_{min}$ or $SP_i=SP_{min}$, last SP_i. is then saved as a new $SP_{trigger}$. Typical detection of the new $SP_{trigger}$ is graphically presented in fig. 4.6. Identified $SP_{trigger}$ value is then passed to On-Off Agent. New $SP_{trigger}$ value is searched after modification of any parameter of the model.

On-Off Agent works similarly to the conventional ON-OFF controller with hysteresis but instead of fixed values at which controller turns ON and OFF this agent uses an adaptive lower boundary equal to $SP_{trigger}$ provided by Tuning Agent.

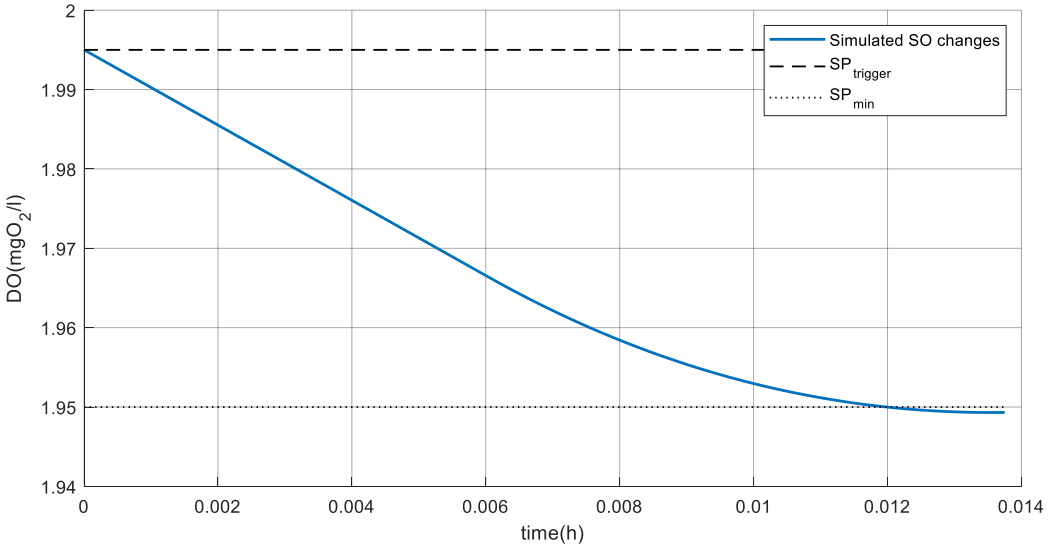

Fig. 4.5 Graphical presentation of aeration trigger value

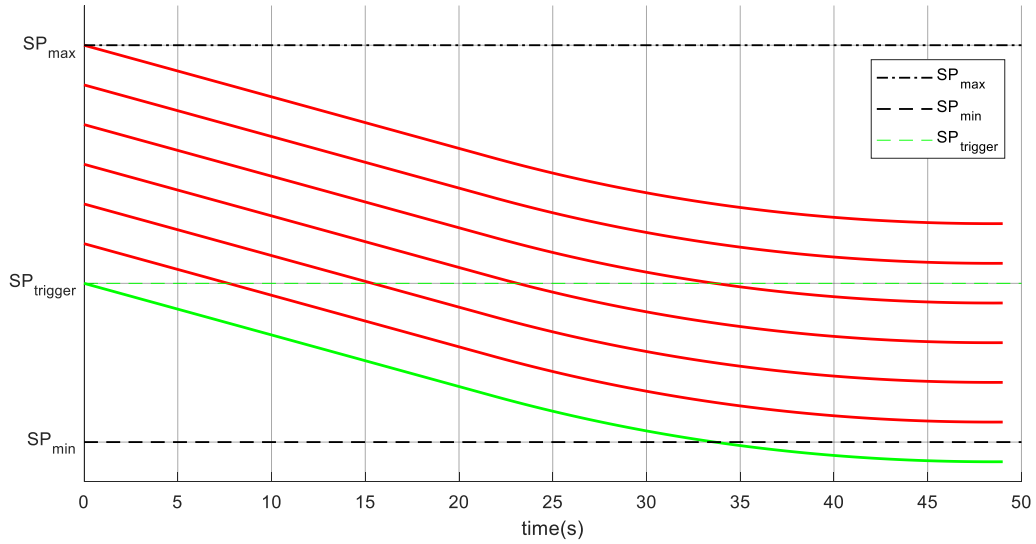

Fig. 4.6 Representation of typical $SP_{trigger}$ tuning process

4.3. Results of experimental validation

Validation is composed of two experiments. The first experiment verifies MAS ability to control dissolved oxygen concentration and its potential to adjust to changing process conditions. Second test compares MAS control performance with the conventional ON-OFF controller. Both experiments are carried out in the laboratory setup presented in fig. 3.1, values of SP_{min} and SP_{max} are constant throughout validation and set at following values, $SP_{min}=1.95(mgO_2/l)$, $SP_{max}=2.05(mgO_2/l)$.

Communication between biological reactor and MAS during experimental validation was carried out with the setup described in fig. 4.7. Initially, agents detect each other's presence in the system using DF catalogue and establish all required relations. This initialization usually takes less than one control cycle, so MAS is able to take control over aeration process immediately after receiving first data from OPC Server. Fig. 4.8 presents control performance of used MAS. During first aeration cycle Conventional On-Off Agent controls the process. In this cycle its competitor cannot provide a reliable control value yet

due to the lack of the estimated process parameters. After first aeration cycle Estimation Agent from controller #2 has collected enough data to identify required parameters and to allow Tuning Agent to tune $SP_{trigger}$ value and begin calculation of control values. As a result, the Selector Agent starts to use its proposed control as it is generated by the agent with higher priority. When control is handled to the controller #2 control quality improves, some undershoots are still present, but they are smaller than ones in the first aeration cycle. After four completed aeration cycles substrate influx into the reactor was modified to induce significant OUR variations. Firstly, substrate influx was doubled, marked in fig. 4.8 with the first green line. During first aeration cycle after the change, the error between minimal DO value and SP_{min} slightly increases, but on the second aeration cycle it is reduced back to the previous level. It suggests that Estimation Agent was not able to update the model parameters in response to substrate flow change for the following aeration cycle. Instead, parameter values identified during lower substrate flow were used which were unable to describe the process that well after the significant disturbance in OUR. As a result, Tuning Agent could not properly update $SP_{trigger}$ for the first aeration cycle after substrate flow disturbance. However, Estimation Agent was able to update model parameters for the next aeration cycle. That allowed Tuning Agent to properly tune $SP_{trigger}$ value and successfully reduce control error. Later, the operator reduced substrate flow back to initial value, which is indicated by the second green line. That did not influence DO control in any visible way, this time agents reacted timely and correctly to the event and tuned themselves by lowering $SP_{trigger}$ value.

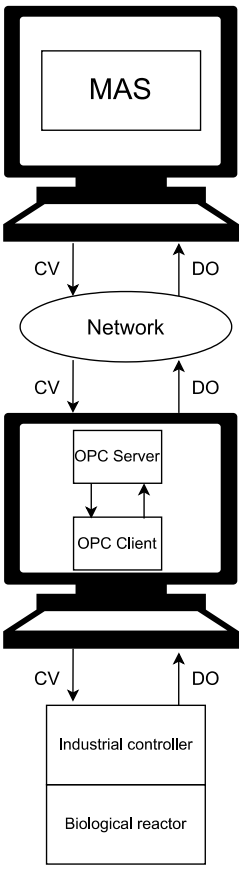

Fig. 4.7 Communication between MAS and biological reactor

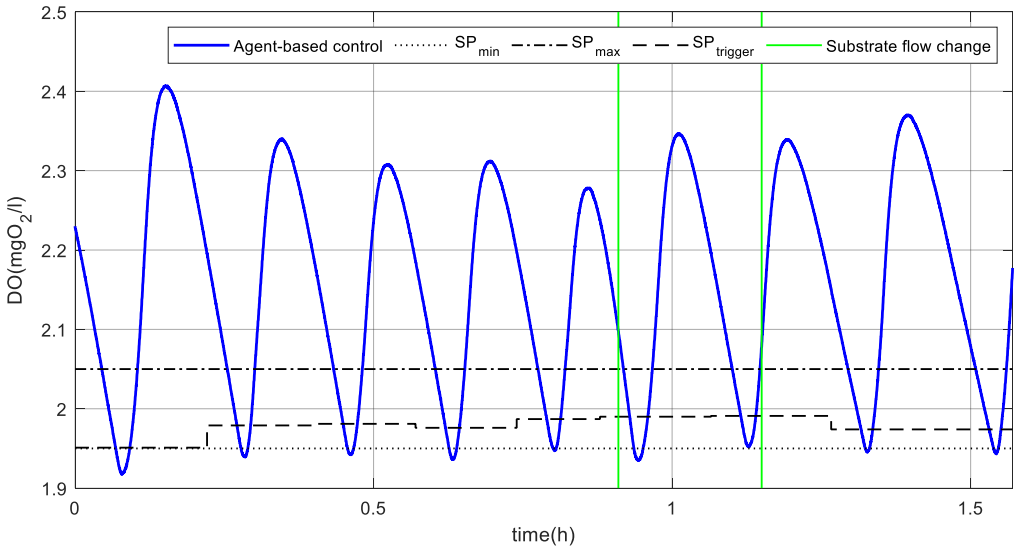

Fig. 4.8 Dissolved oxygen concentration control performance using MAS

Figures 4.9, 4.10 and 4.11 present a comparison of DO control results of proposed MAS with the conventional ON-OFF algorithm implemented in CompactRIO controller. During both measurements substrate flow was kept at constant value. The conventional control, which is presented in fig. 4.9, results in undershoots below SP_{min} of up to 0.05(mgO_2/l). MAS errors, that are shown in fig. 4.10 only slightly exceed SP_{min} boundary and are significantly smaller than ones originating from conventional control algorithm. Comparison of CV trajectory for each controller shows difference in the amount of air pumped into the reactor in every aeration process. Conventional ON-OFF control used 5 pump cycles in one aeration cycle which can be seen in CV graph in fig. 4.9 as five consecutive peaks during aeration phase. Meanwhile for MAS, each aeration cycle presented in fig. 4.10 consisted of 4 pump cycles. This disparity is caused by smaller effective hysteresis gap for MAS controller.

In case of the conventional ON-OFF controller, the hysteresis gap is constant and equal to $H_{ON\text{-}OFF} = SP_{max}\text{-}SP_{min}$. For MAS control H_{MAS} depends on $SP_{trigger}$ value and can be calculated as $H_{MAS} = SP_{max}\text{-}SP_{trigger}$, as described in chapter 4.2 $SP_{trigger} \geq SP_{min}$, thus $H_{MAS} \leq H_{ON\text{-}OFF}$. Smaller gap between starting and ending aeration level for MAS causes that less air needs to be pumped into the reactor to reach satisfactory DO concentration. It is important to note that excessive air dosage during aeration is also energy inefficient according to eq. (3.1) where aeration factor is multiplied by the difference between current DO and saturation concentration. Based on these observations MAS control can be considered more precise and more energy efficient than conventional ON-OFF control.

Although substrate flow to the reactor was identical during both measurements, OUR values slightly varied which appears as a steeper decline of DO concentration for the conventional ON-OFF control. Larger OUR for conventional ON-OFF results in a larger control error which further escalates differences between controllers' performances. However, in author's opinion results shown in fig. 4.11 and results obtained in ideal conditions with identical OURs would not vary significantly and MAS superiority should still be noticeable.

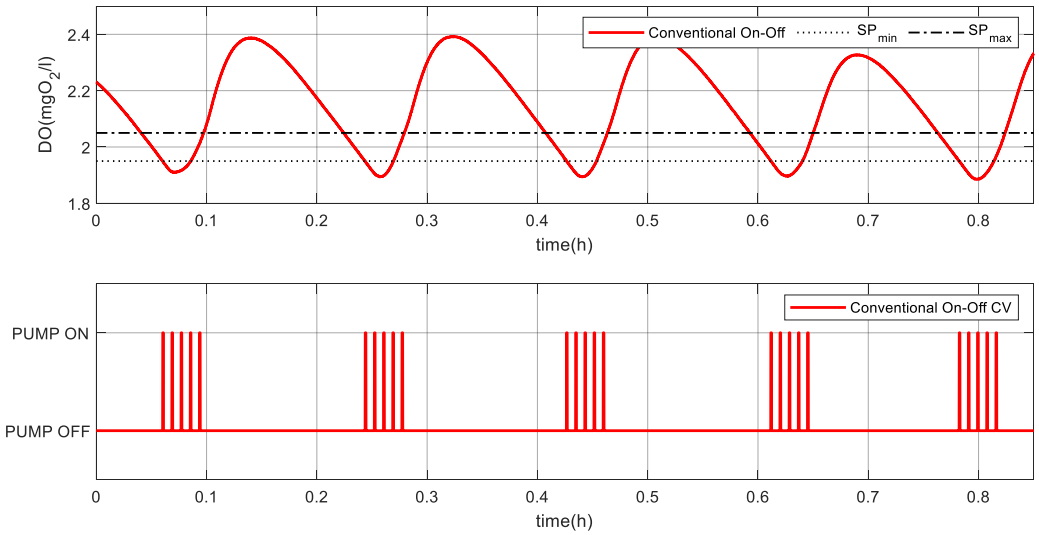

Fig. 4.9 Conventional ON-OFF DO and CV trajectories

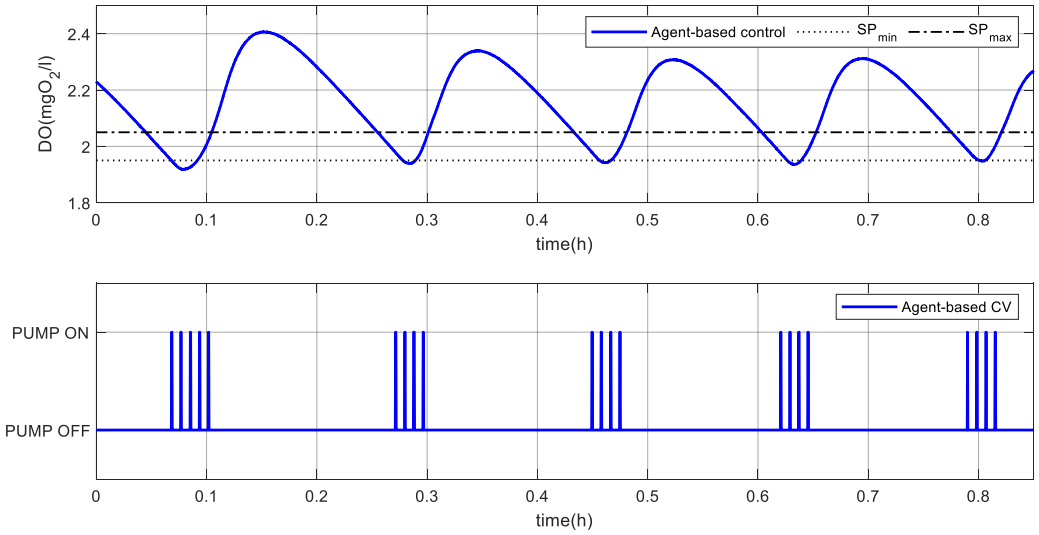

Fig. 4.10 MAS DO and CV trajectories

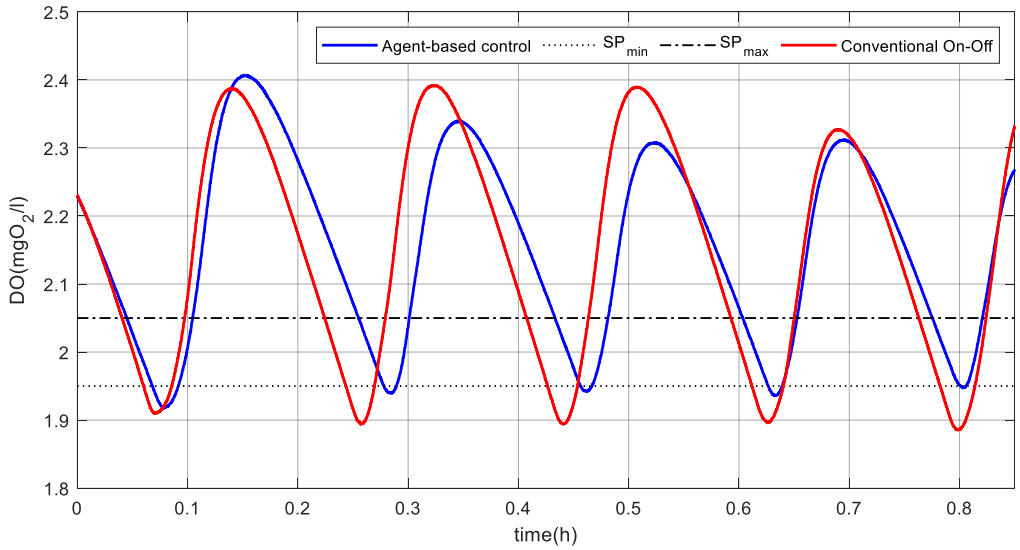

Fig. 4.11 Comparison between MAS and conventional ON-OFF control

4.4. Summary

This chapter presented the multi-agent system designed to control continuous biotechnological processes. Built MAS was incorporating multiple control algorithms and was able to seamlessly switch between them to provide the best available control quality at a time.

Proposed MAS was then tested on DO concentration control in the biological reactor where due to characteristics of the aeration pump, process output could not be kept at a predefined set-point value, so control goal was defined to keep process output between defined upper and lower boundary values. To fulfill defined control goal, a simple adaptive model based control algorithm was designed and implemented in MAS. Proposed control algorithms based on agents visibly improved control quality by being able to fit into the process lower boundary value more precisely than the conventional ON-OFF controller. Due to distributed nature of agents, proposed MAS is additionally able to online tune its control algorithms without influencing ongoing process control, that improves its reliability and ability to adapt to changing process conditions. Dedicated online tuning procedure was developed and described.

Main conclusions from this pilot project in the context of following research for the dissertation are as follows:

- Prepared control system proved that the non-real-time characteristics of agent oriented tools used in the research are sufficient for reliable control of an aeration process with relatively long sampling time. Based on that, decision was made to use JADE together with the same or similar setup like in fig. 4.7 for further research.
- MAS proved its ability to switch control algorithms and adjust to current conditions. However, in this research adjustability was implemented using a simple priority mechanism with only two available alternatives. As a result one of the objectives set for following research was to verify MAS adaptability using higher complexity approaches.
- During this project, only the conventional ON-OFF control and its slightly more complex, adaptive variation were used. It was decided that future research should utilize more advanced controllers optimally including agent specific control algorithms.
- MAS design process encountered many difficulties regarding role assignments of specific agents, their relations with each other, communication methods and form of exchanged messages. This issue confirms statements from previous chapters that development of multi agent control systems lacks a set of common good practices, rules and hints making this process more difficult and time demanding. Providing methods to simplify design of MAS is essential for wider adoption of agent based control systems. Preparation and verification of rules aimed for MAS design simplification is the next objective set for further research.

5. Preparation of rules for efficient development of MAS

After diagnosing key points of interest presented in conclusions of the pilot MAS project for aeration control, research focus was moved to issues causing difficulties in the design and development of multi-agent systems – agents architecture design and the communication between agents. This chapter presents the set of guidelines for developing MAS for a continuous process control that should provide generic advices regarding the most important parts in MAS development. These rules are then verified by developing, according to them, a MAS dedicated to control the dissolved oxygen concentration. This chapter is based on the article presenting results of the investigation published in Przegląd Elektrotechniczny [144].

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. The guidelines presented are generic and focus on the basics of MAS development. Preparation and verification of MAS systems based on validated development rules should provide valuable knowledge in the context of evaluation of usability and feasibility of MAS.

5.1. System architecture design

Figure 5.1 shows a proposition of architecture layout for MAS dedicated for continuous process control. Arrows indicate information flow through the system which together with a controlled process will create a closed loop. The whole system is presented in the form of layers. Each layer is responsible for one part of the control task. Agents are grouped in layers according to their assigned tasks. Every layer should have strictly defined message types that can be received from and sent to the other layers. Agents from one layer can communicate with agents from the other layers using only these signals e.g. design might restrict that agents from Control Algorithm Layer can only receive process values and set points from Input Data Layer and send control values to agents from Actuator Layer. Agents inside one layer should be able to communicate with each other without such strict restrictions, although some limitations might still be applied. Presented architecture is generic and can be used for many types of processes, not only biotechnological.

In the layout, layers in the centre are directly responsible for controlling a process, while Interface Layer and Diagnostic Layer are dedicated to supportive tasks. Such division is based on structure used in conventional control systems which usually realize the following tasks:

- Gathering process' data – this task is realized using sensors that read required parameters. Sensor's data can be then gathered using various approaches e.g. analogue devices might be connected directly to the system input ports that will read their electrical signals values. Digital sensors often present their readings using e.g. a serial port with a dedicated communication protocol and in case of a remotely located sensors, data could be gathered through a network protocol. In the proposed architecture, this task is realized by agents in Communication

Layer that receive data from the controlled process using provided data exchange mechanism.

- Processing gathered data – typical scenario for this task is conversion of 4-20(mA) current sensor signal into a process value. This value is then used to calculate control error that might be used as an input to the control algorithm. In many model-based and predictive algorithms, gathered data is analysed more extensively to derive all required values. This phase should be realized by agents residing in Input Data Layer in the proposed architecture. In general - the task of these agents is to provide to Control Algorithm Layer required parameter values using data gathered from the process as well as information supplied by the user.
- Calculation of a new control value. Usually conventional systems have only one control law implemented that provides control values based on received input. However, MAS properties facilitate development of systems containing multiple cooperating or competing control laws. In the proposed architecture, agents directly engaged in this task should be assigned to Control Algorithm Layer.
- Conversion of control value to actuator signal – derived control value needs to be translated to proper actuator order e.g. a 4-20 (mA) current value or PWM signal. This task should be realized by agents from Actuator Layer.
- Application of the new control to the process – during this task prepared actuator signals are provided to devices. Modes of information exchange are the same as during gathering process data, but data is transferred in an opposite direction. Due to this similarity, that task is also assigned to agents residing in Communication Layer.

Presented layout also includes Interface Layer and Diagnostic Layer. Agents in these layers are not directly dedicated to process control and if needed can communicate with every other agent in the system to perform its tasks. Interface Layer is dedicated for interactions with the user like presentation of control results, receiving new set point values or notification about unexpected events. Diagnostic Layer should contain agents that realize functions related to systems diagnostics and monitoring e.g. fault detection.

If necessary, each layer can be further divided in sublayers as in fig. 5.2 which shows exemplary division of 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 the sensors like a set point or simulation output that may be used by model-based controllers.

Presented architecture layout divides complex systems into smaller, well defined and clearly separated parts which should make them easier to design, develop, maintain and reuse. Each layer forms an almost independent subsystem, thus in case of any modification in layer structure later integration of the changes should 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 [145, 146] and power flow management [45].

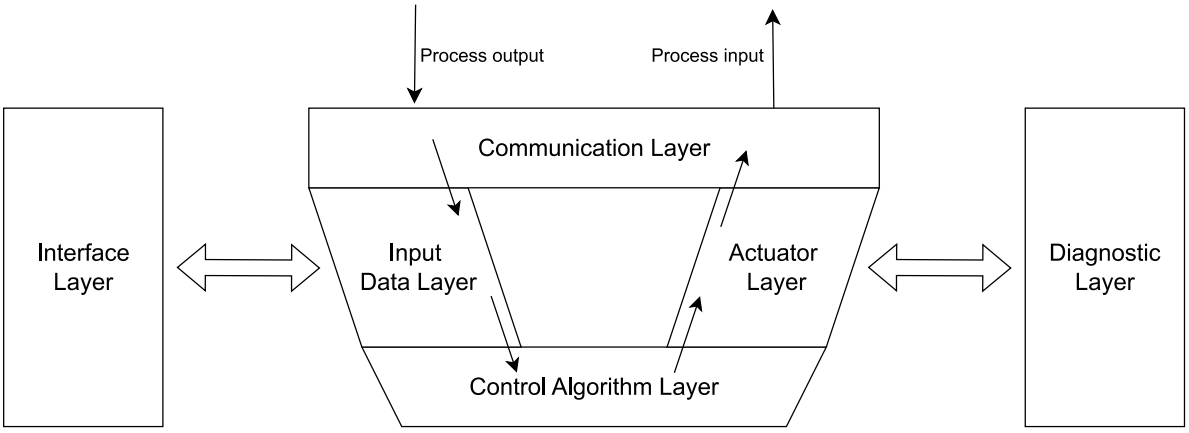

Fig. 5.1 MAS architecture layout

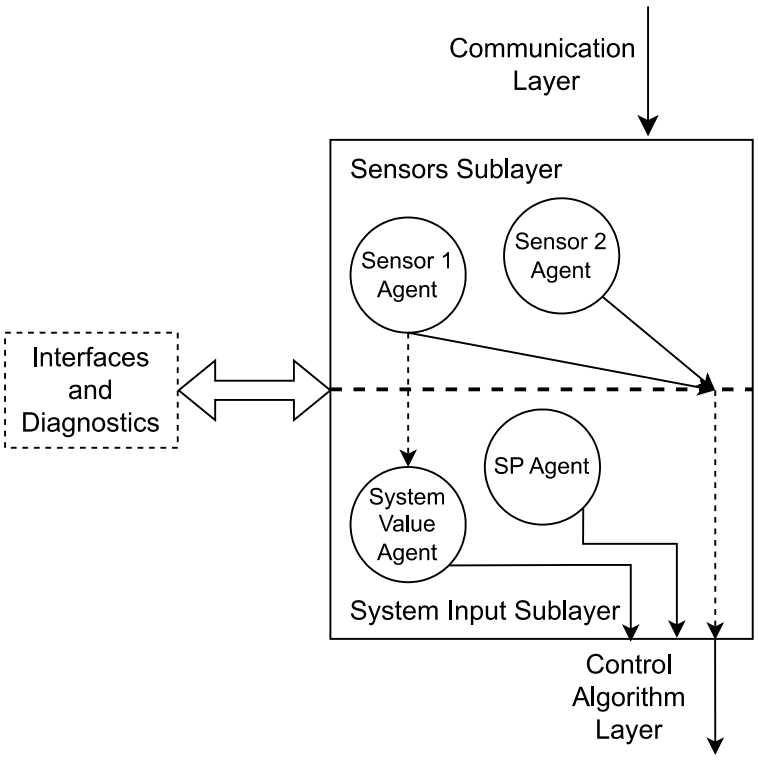

Fig. 5.2 Example of sublayers division

5.2. Agent 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 capable to perform various communication acts, from simple passing of information to the more complex behaviours like subscription mechanism or game theory problems.

Successful communication on any level of complexity requires that agents have shared understanding of content of each other's messages. Such shared understating is also known by the term of ontology. Ontology is a set of statements that defines concepts from a field of knowledge and relationships between them [147]. Using ontology in information exchange ensures that agents have a 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 the prepared multi-agent 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 a broad area. Process control systems usually require concepts that are only a small parts of these knowledge domains i.e. knowledge about a 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 schemas, fig. 5.3 presents a proposition of base layout of ontology schema for agent-based control systems for continuous processes.

The schema is based on grouping all concepts into three areas. Division into three areas is based on high level view on agent's functionality. In general, agents are performing actions. Performing any action might require additional data. Doing the action might also produce additional information as a result. Based on that, the ontology domain was split into 3 subdomains. *Actions* is a set containing descriptions of agent's actions, *Data* and *Diagnostic* hold descriptions of data concepts. *Data* area contains classes that represent all instances of data that agents exchange during process control tasks. Hierarchy inside this area shows an example of base class organization. This hierarchy might be sufficient for systems using simple control algorithms like PID, but for more complex systems this schema would need to be extended according to an individual needs. *Diagnostic* area contains classes describing concepts that are not directly connected to the process control but are used in MAS side tasks e.g. diagnostics or access control. Classes inside *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 [148]). In such cases importing this set of actions directly into the ontology might be the most efficient approach. 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 an ontology according to these guidelines results in only few of such interactions.

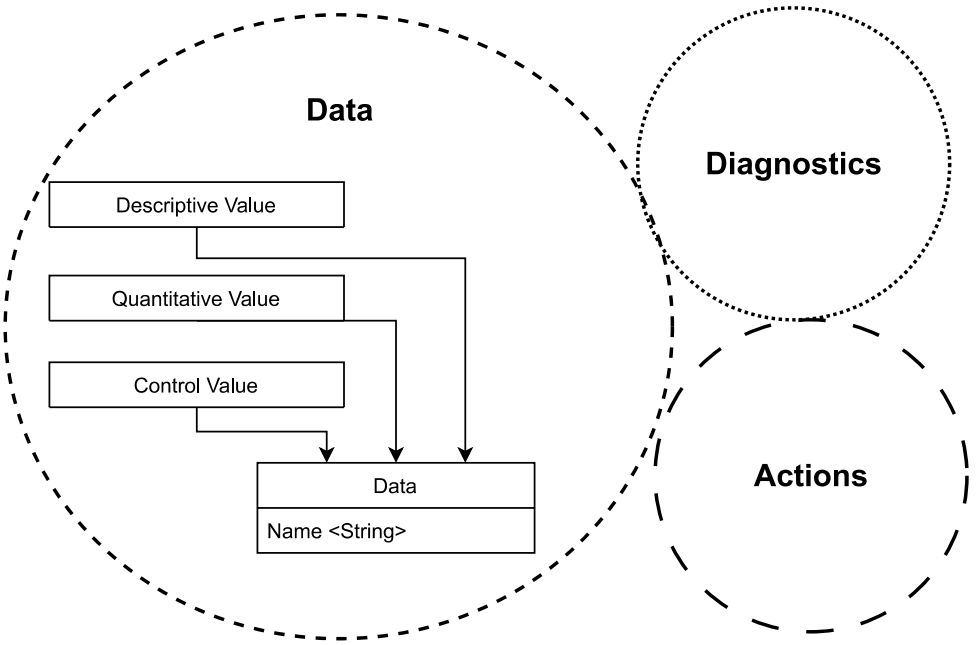

Fig. 5.3 Proposed ontology layout with example of Data area organization

5.3. Verification of rules

In order to verify the presented rules, MAS is built according to those guidelines and its functionality is tested by using the system to control dissolved oxygen concentration inside the simulated bioreactor. 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 concentration. In this experiment, due to mutual similarity and overall simplicity of competing controllers no explicit switching mechanism was implemented. Instead, moment of switch is based on instruction provided by the user through the user interface. Switching between controlling agents will be performed seamlessly.

MAS developed according to the rules is presented in fig. 5.4. *Communication Layer* contains one *OPC UA Agent* that exchanges data with the process through OPC UA protocol connecting MAS with the simulation of the aeration process as presented in fig. 5.5. *Input Data Layer* is populated by three agents, all of them provide data to *Control Algorithm Layer* as presented on the diagram. *Last CV Agent* converts the simulated 4-20 (mA) signal that controls the aeration pump into 0-100% range used by controllers. *SP Agent* reads SP value from its user interface and *DO Agent* decodes new DO values from plant messages and sends them further. *Control Algorithm Layer* contains *Selection Agent* that reads information from its user interface to detect which *PI Agent* should be selected. That information is sent 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.

Created MAS uses ontology that is prepared based on the layout from fig. 5.3 and is presented on fig. 5.6. This ontology can be treated as a proof of concept, based on that, scaled-up version was prepared, which is presented and described in details in the next chapter. Diagnostic domain is empty because, as mentioned previously, this MAS does not contain diagnostic functionalities.

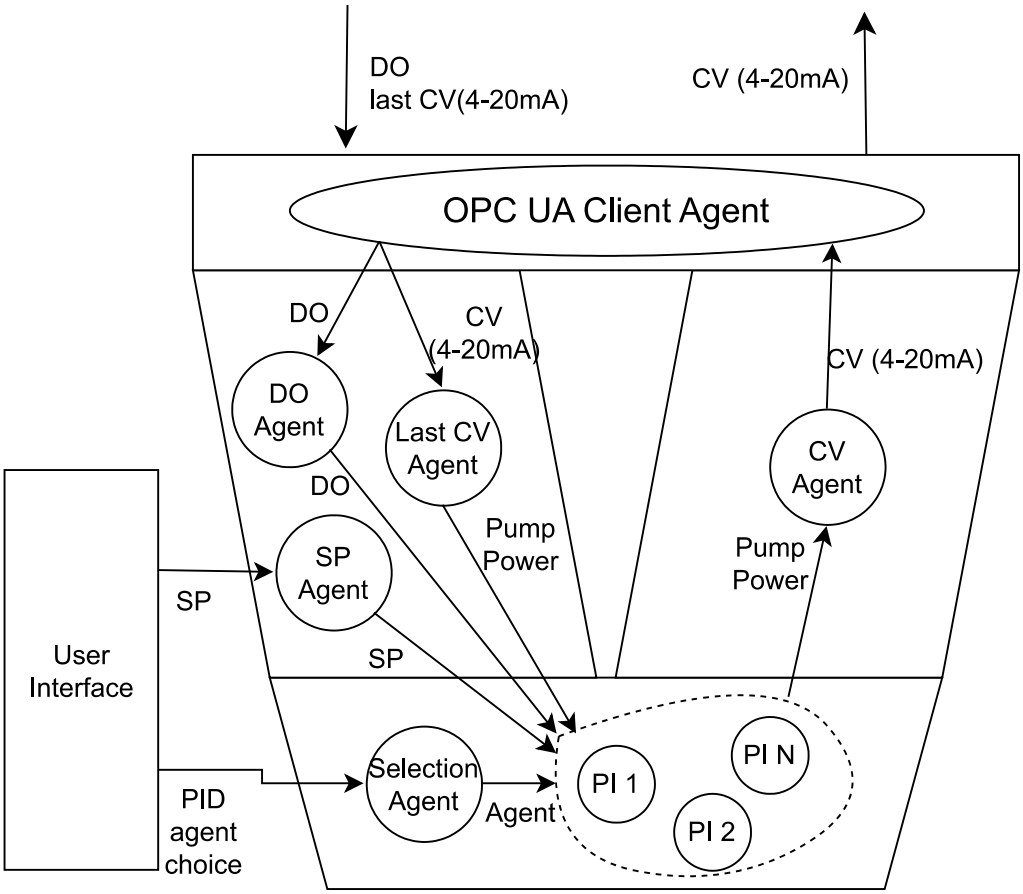

Fig. 5.4 MAS architecture for the verification experiment

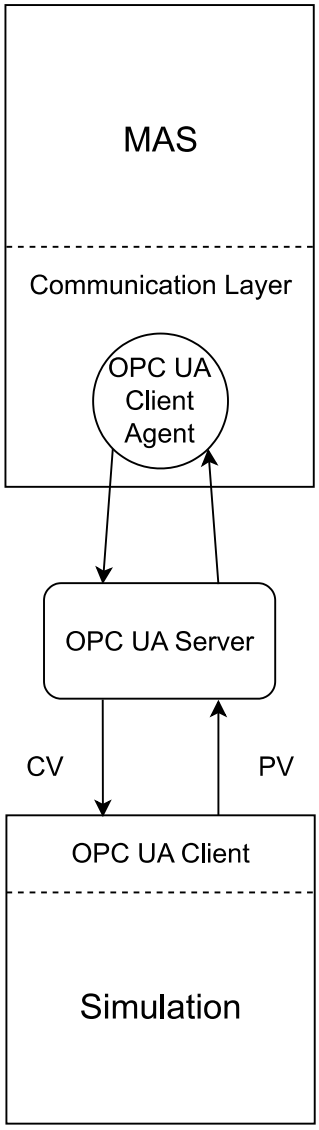

Fig. 5.5 Setup for the verification experiment

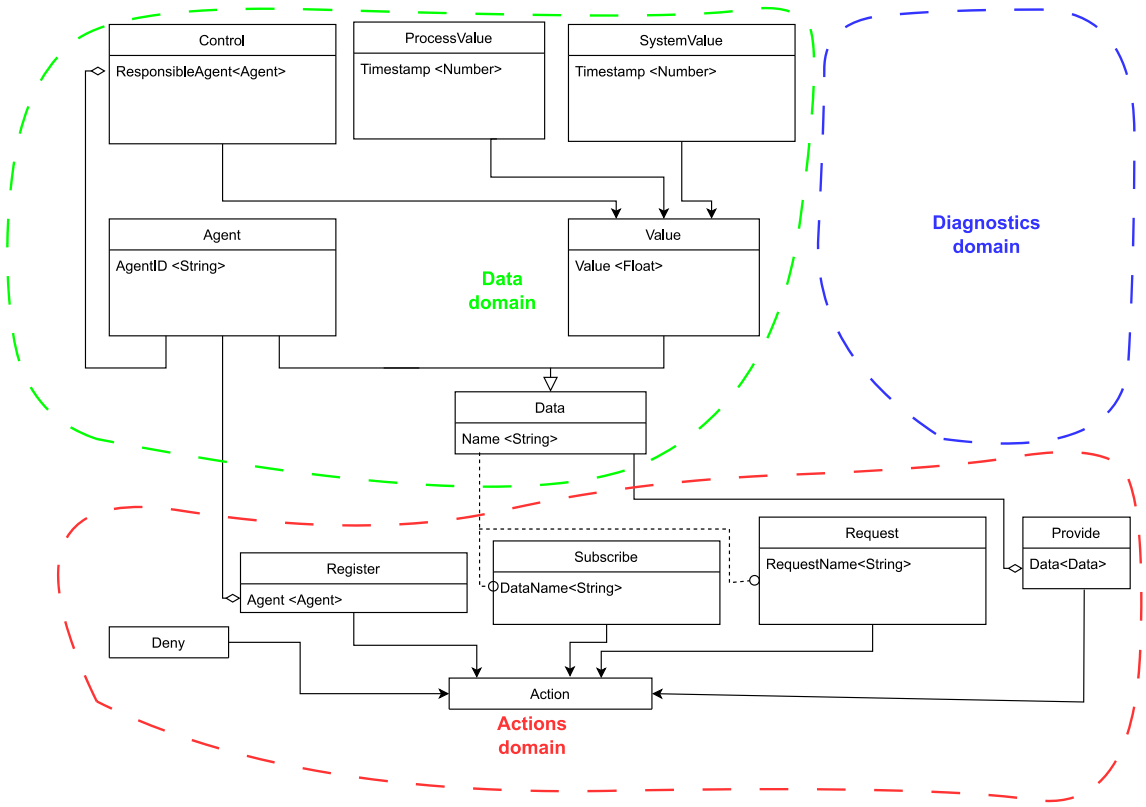

Fig. 5.6 MAS ontology for the verification experiment

Table 5.1 PI agents tunings for experiments

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

Simulated aeration process together with its simulation implementation is represented in chapter 3. DO_{sat} (mg O_2/l) value is set at 10 (mg O_2/l), *OUR* (mg O_2/lh) for the time of the experiments is considered constant and equal to 11.88 (mg O_2/lh).

Validation included three experiments that used two PI agents. Tunings for PI agents were derived using Chien-Hrones-Reswick procedures. Firstly, a first order plus dead time model of the aeration process around the set-point of 2.5 (mg O_2/l) was identified. Then, using identified parameters values, PI 1 was tuned according to "aggressive" rules and tunings for PI 2 are based on "conservative" rules. Resulting tuning values are presented in table 5.1.

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. For the last experiment, at the beginning control was handled by PI 1 agent, but when DO value exceeded 10% of the SP step size (i.e. rose above 2.05 (mg O_2/l)) control was switched to PI 2 agent.

Result of the experiments are presented in fig. 5.7, 5.8 and table 5.2. 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.

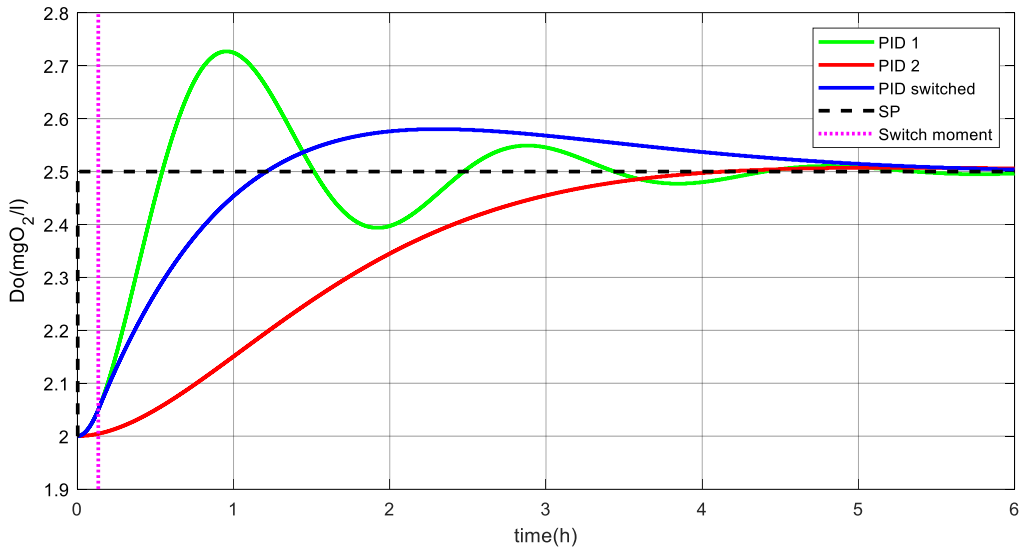

Fig. 5.7 DO trajectories during experiments

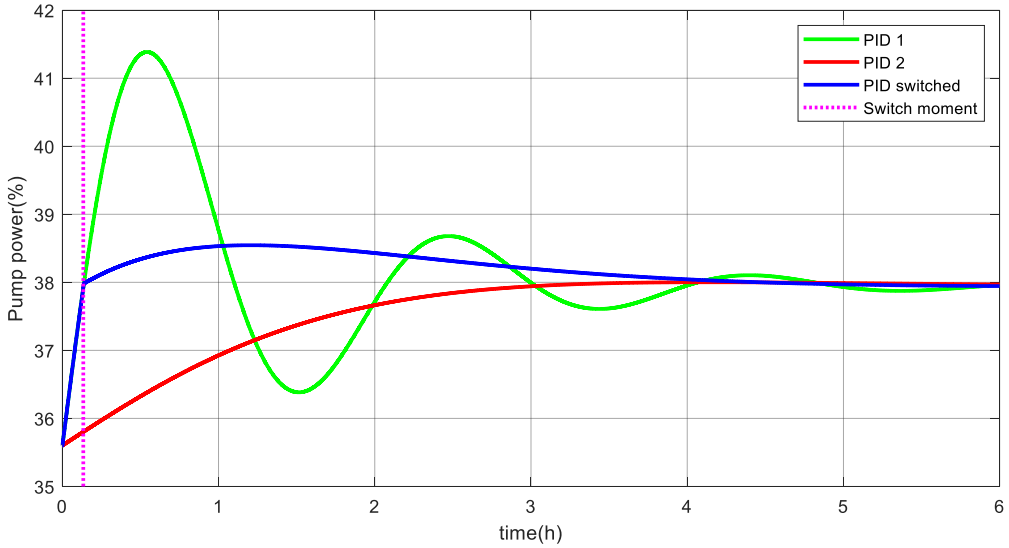

Fig. 5.8 Control value trajectory during experiments

Table 5.2 Control quality indicators values for each experiment

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

5.4. Summary

This chapter presented the set of rules for development of MAS for a continuous process control. Presented guidelines are generic and focused on the basics of MAS development. Such guidelines might provide assistance for dealing with the issues that slow down the development of MAS in the continuous process control area like unfamiliarity with agent-based approach and lack of existing good practices. 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 a system is divided into layers, every system layer is then described and explained. Communication rules present the set of tips to use during ontology preparation and the proposed base of the layout that can be used during development of a new, dedicated ontology.

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

On the other hand, presented experimental results proved that MAS adjusting mechanism even in the form of simple threshold mechanism might improve the overall control quality.

The result of this research – sets of rules for MAS architecture design and ontology preparation is a base for following research projects presented in this dissertation. Later work focus on scaling up solutions based on these approaches and using them on more complex systems with more sophisticated control algorithms.

6. Ontology in MAS for control of biotechnological processes

Previous chapter introduced key challenges in development of multi-agent control systems – lack of common rules regarding MAS architecture design and preparation of communication mechanism between agents. This chapter focuses more deeply on the second of presented issues.

Unlike object-oriented programming, where inter-object data exchange is tightly coupled with relationships between them, agents are more independent software constructs which relations with their environment are not that strict. These relaxed connections between agents cause, that direct data exchange from one agent to another like in object oriented or functional programming (e.g. passing data through method or function arguments) cannot be implemented. Instead, information in inter-agent communication is wrapped in higher level messages and transported between agents using environment specific mechanisms. This approach requires that an agent is capable of understanding received messages syntactically and conceptually.

An agent able to understand message syntax is aware of its structure which, besides actual information, may also contain additional metadata e.g. sender ID, time of sent, transport protocol etc. Message structure compatibility is in most cases resolved by using well known, standardized message structures e.g. FIPA-ACL Message Structure [143] prepared specifically for agent-based applications, MAS might also use non agent specific message exchange approaches like MQTT protocol [149] as long as both ends of information exchange are aware of chosen messaging mechanism.

Besides syntax rules applicable to messages structure, the recipient of a message needs to also be aware of syntax of the message content i.e. part of the message containing information that sender wanted to pass. Message structure specifications described in the previous paragraph usually do not specify format of message's content, so it needs to be agreed upon separately. However, in most cases there is no need to design case specific message content formatting and some well-known, already established data encoding format might be used, e.g. XML, JSON or YAML which syntaxes are both machine and human readable.

To sum up, message syntax compatibility between agents inside MAS can be achieved using well known, general purpose tools and standards, for which proficiency in agent-based systems development is not needed. However, efficient communication between agents requires common conceptual understanding of message content as well. Conceptual understanding means that message sender and recipient share a knowledge base and understand content of message in the same way, e.g. both receiving and sending agent use "SP" as an abbreviation for control set point.

Agents' shared knowledge base is usually built using ontologies. The term "ontology" originates from philosophy and was later adopted by computer scientists [150]. Multiple definitions of the ontology can be found in literature. Probably the most prevalent one states that ontology is a formal representation of concepts of a domain knowledge that is used to share, reuse and analyze domain knowledge and to enforce common understanding of the structure of information among interested parties [150-152].

Multi-agent systems rely on efficient communication between agents. Therefore, ontologies are commonly used in MAS to ensure that agents inside the MAS have shared knowledge base. Some sources advise that the ontology should be prepared before the MAS development, because it defines a knowledge domain that the MAS will be based on [152]. Many examples of ontologies used alongside MAS, from broad range of disciplines can be found in the literature. Ontology based MAS are used in medical research for e.g. preparing assistance for patients under rehabilitation [153], evaluating health aspects of people's diet [154] or collecting, sharing, integrating and managing information about human diseases from multiple information resources [155]. Many use cases can be also found in other areas not directly connected with the automatic control. One of these areas is education where this type of systems were evaluated as a support tool used in learning processes [156] and system designed to increase management capabilities in "smart schools" [157]. For the business management, MAS paired with ontology is used to create virtual collaboration platforms for multiple enterprises [158] or integrate enterprises' internal data stores with information found in relevant web sources [159]. MAS was also used alongside a large scale OASIS ontology to create blockchain oriented e-commerce [160]. In urban organization, these systems serve as tools for coordinating work of services in a city [161] or as an environment used for simulating urban freight transportation [162]. Other uses cases include support systems for air traffic management [163], cybersecurity focused MAS that implements outbound intrusion detection [164] and systems to support software development from multiple remote sites [165].

Ontology based MAS also found increasing interest in automatic control related areas, especially in energy, smart grid management and manufacturing, mostly as decision support systems used in e.g. automation software for smart grids using a semantic web ontology [166], power system management that uses the ontology to enhance its interoperability with other heterogeneous systems [167] or a prefabricated component supply chain [168]. These systems were also verified by working as systems for solving conflicting transaction and scheduling problems among supply chain members using negotiations [169], tools dedicated to control a resource management system [170], or as systems to improve the overall equipment effectiveness of shop floor machines [171]. These areas mostly match general main areas of interest of MAS usage, that were described in previous parts of this dissertation. Similarly, use cases of ontology based MAS in the process control area are noticeably less common. However, some research works for various types of processes can still be found. MAS designed for a collaborative process control that uses an ontology to formalize interface between OPC communication protocol and JADE constructs is described in [172]. Process supervision in a large-scale chemical plant that incorporates an ontology as a source of formal classification of process descriptors is presented in [91] and an expert system for validation of a hybrid control of biotechnological plant can be found in [173].

In this chapter a generic ontology schema designed for MAS for process control, especially for biotechnological processes is presented. This schema is an upgraded version of ontology prototype from the previous chapter and is prepared as a part of the set of rules dedicated for efficient development of multi-agent systems.

In the present, large choice of publicly available ontology repositories exists (e. g. Ontolingua [174] or DAML [175]). These ontologies are often designed to be generic and suitable to reuse in wide aspect of projects, in many cases it is highly advisable to browse these archives and adapt one of existing ontologies instead of sacrificing time and effort in creating a new schema from scratch. According to author's observations, these ontologies are usually dedicated to large scale systems where common understanding of a broad range of concepts is required. In these types of projects, reusing existing ontologies provides highest benefits. However, such ontologies do often form large, complex schemas and require additional tools like ontology engines to integrate into a project to be able to use them. For process control tasks however, such large ontologies are usually not required. Systems dedicated to control plants in real-time usually need knowledge from the area limited to characteristics of controlled process, actuators, sensors and external factors that may influence the control quality. Based on that, author assessed that usage of available tools for ontology management would cause unnecessary complexity and decided that the designed ontology should not rely on any available ontology engine and instead use more lightweight tools to manage the generated knowledge base. As a result, author decided that the main objective is to design a generic schema with relatively small number of classes that do not require dedicated full scale ontology engine but are able to describe concepts in the area of interest with satisfactory precision.

6.1. Ontology schema

Layout of the prepared ontology is presented in figures 6.1, and 6.2. To make visual representation of ontology easier to understand, it is divided into three groups called domains, each domain contains classes dedicated to specific purposes.

- Data domain presented in fig. 6.1, can be considered as a main domain of the ontology. Classes inside this group represent concepts of information used by agents to perform their actions. The root class for most of concepts in the domain is a class called *Data* which is a generic representation of information and contains information about their name. In the presented layout this root class has two descendants, one of which is called *Value* and is an abstraction that represents any concept possible to describe with a single numeric value. The class Value has three descendants: *ProcessValue* represents all the values gathered directly from a process (i.e. sensor readings), the *SystemValue* class describes values that are not gathered by the sensors but derived from *ProcessValues* through some type of calculations. The last descendant is the *Control* class which is a concept dedicated for describing control values. Described idea of data and value concepts hierarchy is in author's opinion universal and can be reused in applications that adopt many other control systems and algorithms. Data domain contains also *ControlOffer* class that represents concepts used when multiple control algorithms reside in the system and MAS should choose the best one. *ControlOffer* instances should contain information about a proposed control value and an additional value that would allow to choose the best control offer according to a chosen criterion. The last two concepts presented in data domain are *Actuator* that represents a real

actuator in the system. This concept might be used in cases, where there are multiple actuators in the system which would require control values to specify the applicable actuator. *Agent* concept is self-explanatory – it is dedicated to describe agents.

- Actions domain visible in fig. 6.2, is based on FIPA Communicative Act Library Specification [148] with the addition of new and modification of already presented concepts. This domain contains classes that represent interactions between agents with *Action* as a root class. *CallForOffers*, *Offer*, *Subscribe* and *Request* are classes that describe different stages and methods to gather data by agents. Using *Subscribe* to get data results in receiving messages with specified information periodically (most likely in *Producer-Consumer* relation). On the other hand, *Request* is used to get data only once (like in *Client-Server* relation). Information (i.e. instances of classes from data domain) might be exchanged between agents through *Provide* action. *Register* action is used to inform other agents about the presence of a new agent in the system and *Deny* represents a negative response to agents' requests.
- Diagnostics domain at this stage is intentionally left empty, the purpose of this domain is to group concepts that are used by additional functions (e.g. fault detection algorithm) implemented in MAS which are neither closely related to project control nor describe actions taken by agents.

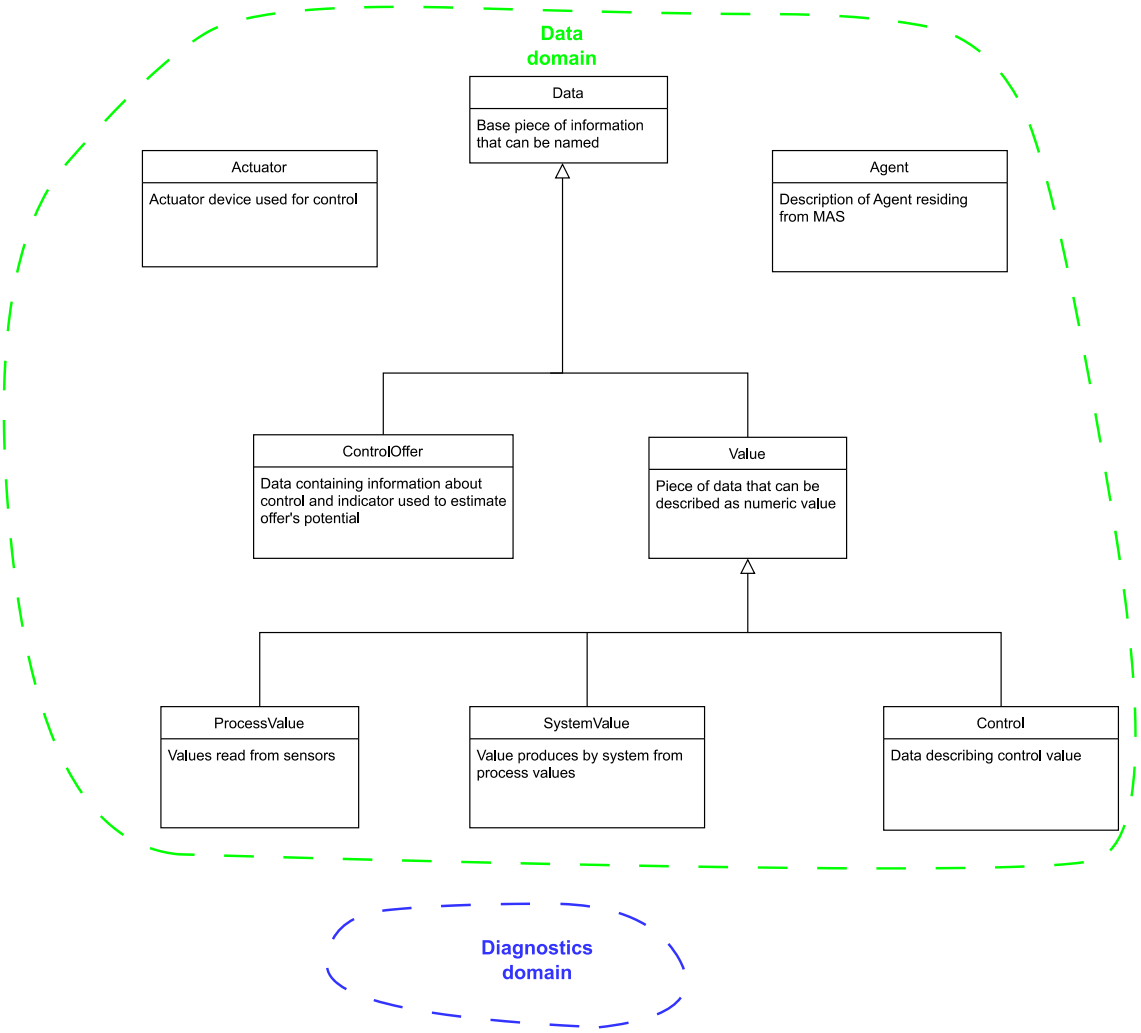

Fig. 6.1 Ontology layout – Data and Diagnostic domains

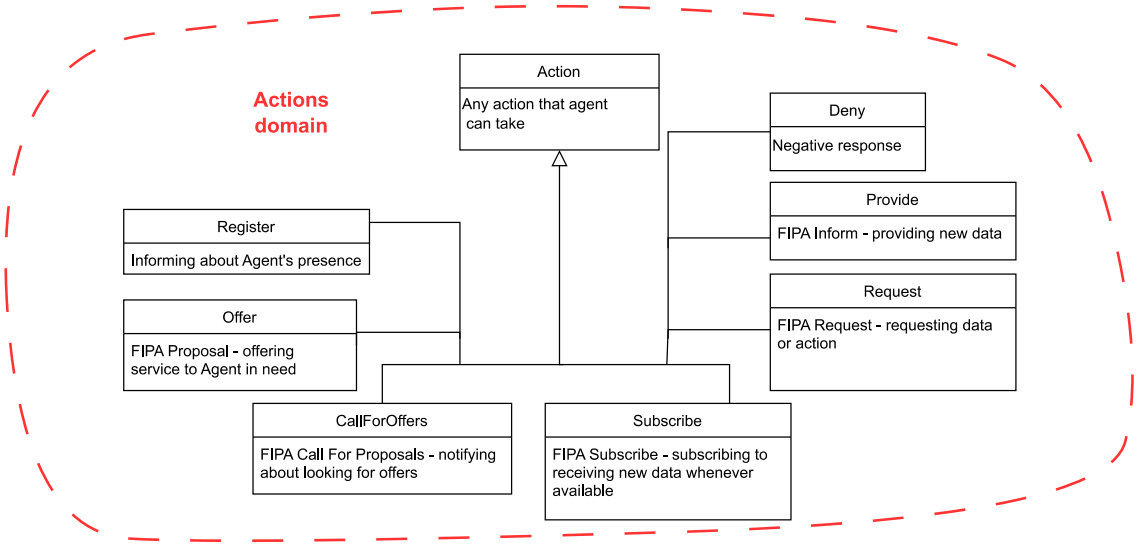

Fig. 6.2 Ontology layout – Actions domain

6.2. Object model based on ontology.

Object model [176] created using the previously described ontology schema is presented as UML diagram in fig. 6.3. This model has been implemented in Java and is used as a base for MAS ontology systems in experiments described in following chapters.

Information in the form of instances of classes from the presented object model can be passed between agents in messages as a XML strings which are serialized from and can be deserialized back to a Java objects. For objects serialization, a tool called Jackson is used. It is worth to note that JADE also contains a built in parser to interpret ontological concepts from XML strings.

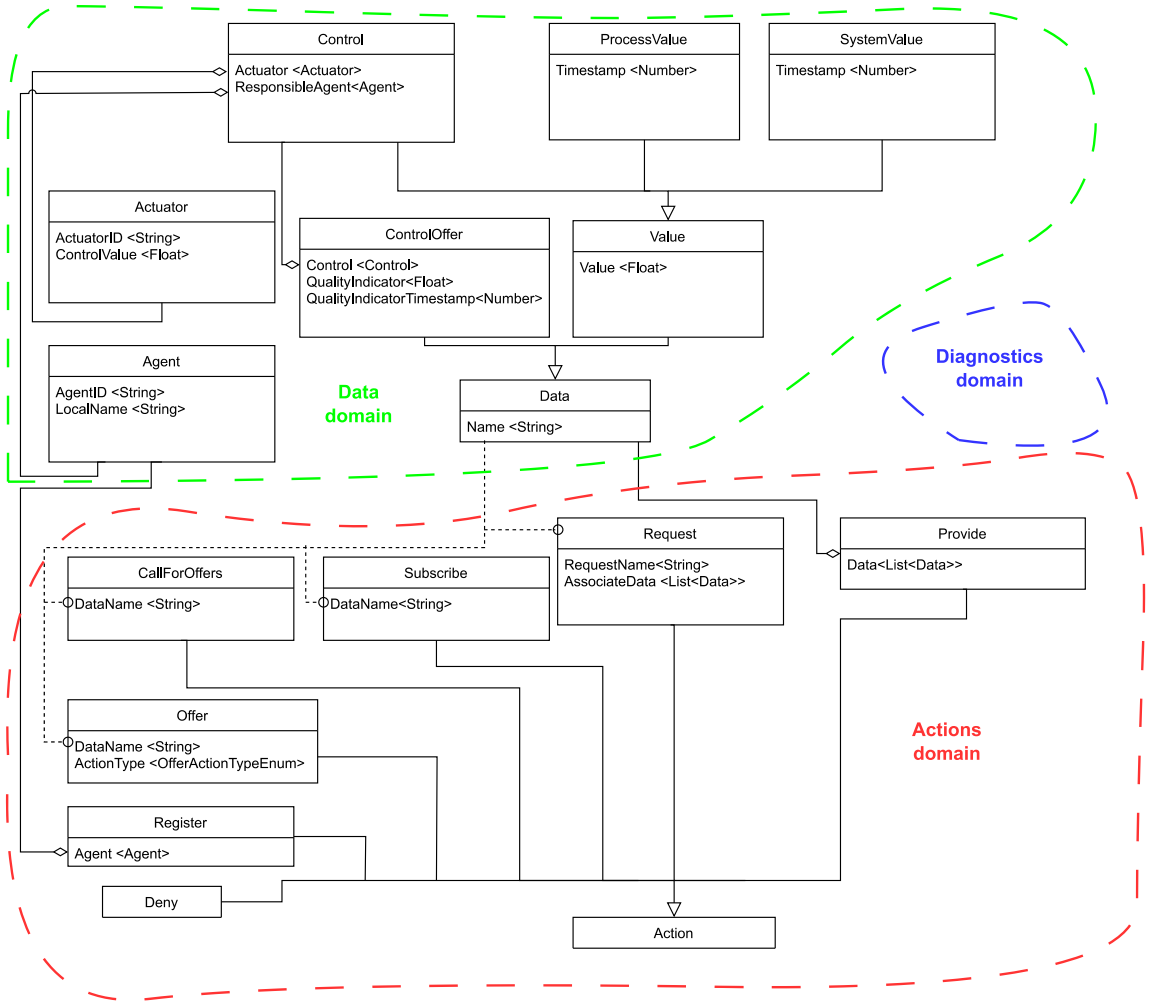

Fig. 6.3 Object model based on ontology

6.3. Summary

Presented ontology fulfills the main objective set before the design. The data and value concepts hierarchy form an universal core of prepared ontology that can be reused in other control systems. Other classes can be easily modified, added or removed to comply with specific needs of any application. Additionally, the number of classes used in ontology is still low enough to allow relatively simple manual modifications without a need to use any large scale tools dedicated for ontologies.

7. MAS for continuous control and its application to activated sludge process

In this phase of the doctoral research, presented in previous chapters rules of MAS development alongside the proposed ontology were tested in development of control system for a real process plant. The task is to prepare and verify a MAS for a dissolved oxygen concentration control in the activated sludge wastewater treatment process in a multiple-input control setup. The system should give the ability to alter MAS configuration during runtime through additions and removals of specific agents. The main points of the research are as follows:

- Application of MAS which uses a dynamical model of DO concentration, prediction algorithms and multi-agent social interactions to operate multiple aeration pumps with various characteristics used to keep the process value precisely within a desired range.
- Presentation of an evaluation mechanism for agents, which is used as an indication tool to find agents, whose calculation results do not match real process measurements.
- Introduction of a fault detection algorithm that uses the provided knowledge about the system to detect malfunctioning agents.
- Example of a practical implementation of described multi-agent system with an evaluation mechanism and a fault detection algorithm included.
- Experimental validation during aeration of a real, laboratory scale setup of activated sludge process.

This chapter is based on the article published in the Bulletin of the Polish Academy of Sciences[133].

7.1. Definition of control goal

Control goal for the proposed system is to keep the DO concentration precisely within the desired range of [SP_{min}, SP_{max}]. Control values should be derived by dedicated agents and supplementary data, e.g. information about model parameters should be provided by separate agents. For this specific task of DO concentration control MAS CV calculations should be based on the BBPC algorithm. Process controlled by the MAS needs to reach both its lower value boundary SP_{min} and its upper value boundary SP_{max} with similar precision. However, as mentioned in chapter 4 and [26], single input system based on an ON-OFF pump and the BBPC control algorithm are not able to reliably fulfill that requirement. To achieve that, a smaller variable speed pump was added to create the multiple-input control system as shown in fig. 7.1. Similar multiple blower systems composed of both ON-OFF and variable speed pumps are also used in large scale wastewater treatment plants (WWTPs) [30]. Due to the reasons described in chapter 3.1.1, the ON-OFF pump can be turned ON only for a fixed time during aeration cycle. Due to this technical limitation, both pumps operate in 30(s) aeration cycles. Variable speed pump can work with shorter aeration cycle times, but due to slow dynamics of the aeration process, reduction of aeration cycle time would not provide significant improvements, hence author decided to use identical aeration cycle times for both pumps. The designed

MAS needs to operate both aeration pumps to reach boundaries as close as possible. Additionally, aeration should be mainly performed by the ON-OFF pump due to its higher efficiency and the variable speed pump should be used only for reduction of a remaining control error. In order to avoid an another layer of complexity to the control problem regarding complex dynamics of air flow, an assumption is made that both pumps cannot be in the ON state simultaneously.

$$\frac{dY(t)}{dt} = f(\, Y(t), CV(t)\,) \tag{7.1}$$

$$\hat{Y}_{i+j} = \hat{Y}_{i+j-1} + \Delta t \cdot f\big(\, \hat{Y}_{i+j-1}, CV = ON \,\big),\ j = 1 \dots H \tag{7.2}$$

7.1.1. Boundary-Based Predictive Controller

Boundary-based predictive controller is dedicated for processes that use ON-OFF actuators to control the process output in an oscillatory manner. Control goal of BBPC is the same as the MAS control goal for this chapter i.e. keeping output value within the desired range of [SP_{min}, SP_{max}].

Unlike conventional ON-OFF controller, where control is changed only when process output goes out of [SP_{min}, SP_{max}] range, BBPC tries to constantly keep process values inside provided range using consecutive predictions of future process behavior, assuming that control is changed at the current moment. To efficiently predict process behavior, its model needs to be known. In general, the process model used by the BBPC can be described in a form presented in eq. (7.1) where *Y* is the process output and *CV* is a state of the actuator (ON or OFF). Accuracy of provided process model determines how well BBPC will be able to meet its control goal.

Prediction of future behavior of *Y* is performed at every sampling moment t_i and its current value is used as an initial condition $\hat{Y}_i = Y(t_i)$ for the prediction procedure. Assuming that initially *CV=OFF* prediction algorithm can be described with eq. (3.2), where Δt is a sampling time and *H* is a varying prediction horizon for which $\hat{Y}_H = \hat{Y}_{min}$. The control will be switched when $\hat{Y}_{min} \leq SP_{min}$. After CV is changed to ON, the prediction is performed in the same way, but this time with the assumption that CV is immediately switched to OFF. Actual switch occurs when predicted maximal process value $\hat{Y}_{max} \geq SP_{max}$.

Concept of BBPC alongside its detailed description and tests on real test plant can be found in the article published in the IEEE Transactions on Industrial Electronics [26], which is co-authored by the author of the dissertation.

For this chapter, BBPC uses model (3.1) to predict future DO concentration changes. To ensure satisfactory control quality, MAS needs to be able to precisely identify values of process's parameters.

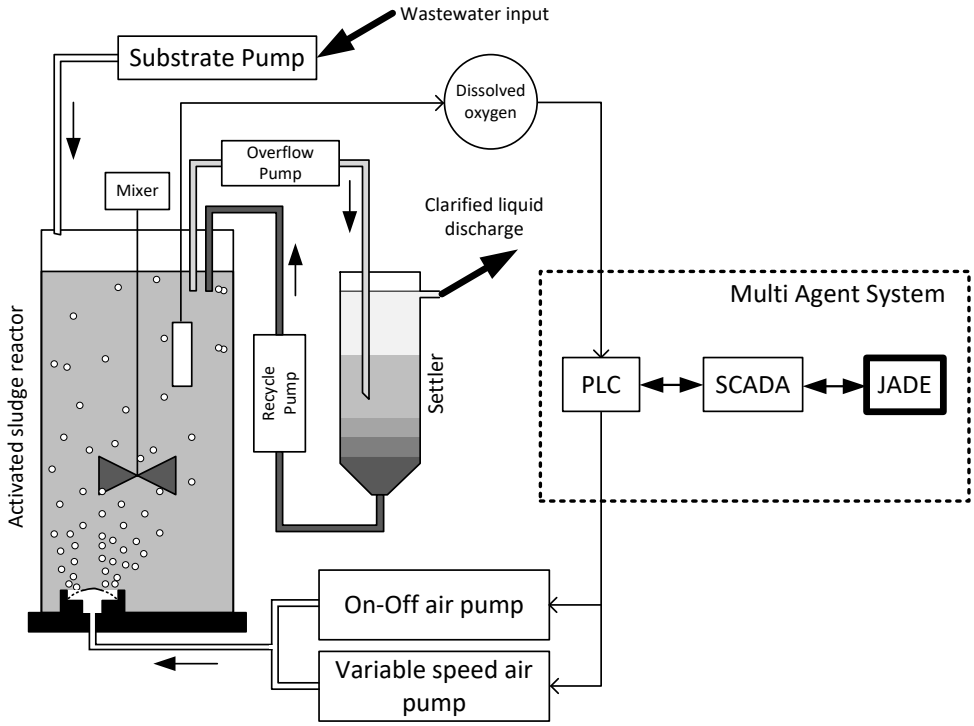

Fig. 7.1 Schematic diagram of DO concentration control system

7.2. MAS implementation architecture

Architecture of the system is based on rules described in chapter 5 and its layered view is presented in fig. 7.2. As can be seen, the prepared implementation follows the layout and data flow forms a loop with *Communication Layer* being a gateway for both incoming and outgoing process related values. However, some deviations from the rules are present, that can be noticed by analysing the flow of *Process values, Emergency control* and *CV* in fig. 7.2, *Process values* are passed from *Communication Layer* straight to *Control Algorithm Layer* omitting *Input Data Layer* and *Emergency control* is transferred from *Control Algorithm Layer* ignoring *Actuator Layer*. Lastly, *CV* is fed back to *Input Data Layer* and *Control Algorithm Layer* ignoring system's *Communication Layer*. In author's opinion this case does not actually break the proposed rules. Flow for these values could be redrawn in a way that they would flow through omitted layers without interaction with any of the agents, making the data exchange layout compliant with the rules.

$$DO_{diff} = \begin{cases} \left|SP_{max} - \hat{Y}_{max}\right|;\ CV = ON \\ \left|SP_{min} - \hat{Y}_{min}\right|;\ CV = OFF \end{cases} \tag{7.3}$$

Practical implementation of the proposed multi-agent system architecture for the DO concentration control was created using JADE framework. Every layer contains agents realizing the following tasks:

- *Communication Layer* - contains a single agent that operates OPC UA client, that is based on an open-source implementation of OPC UA communication protocol called Eclipse Milo. For DO concentration control only one *process value* is used – current DO concentration reading from the dissolved oxygen

probe. Agents can either subscribe to the agent from *Communication Layer* to periodically receive up to date DO values or request to provide only current DO. Additionally, the subsystem sends to OPC UA server *CV* values that are prepared by the MAS.

- *Input Data Layer* – fig. 7.3 presents schema of this layer, it consists of two agents that derive two system values, one is $k_{La}(t)$ that is calculated by k_La *Agent* using transformed (3.1b). Another *system value* is *OUR(t)*. Although *OUR(t)* directly describes oxygen uptake rate of controlled process, its measurements are not available in this case, thus its value is estimated by *OUR Agent* from a declining DO slope using methods described in [26]. These agents are sending their calculated values on-demand – when the appropriate request is received.
- *Control Algorithm Layer* – consists of five agents organized in a way presented in fig. 7.4. k_La *Prediction Agent* is responsible to perform calculations based on owned k_La model and provide a predicted k_La response to received u_{air}. *BBPC Control Agent* calculates control offers based on the ON-OFF pump using the BBPC control law described in [26]. *Peristaltic Pump Agent* generates control offers based on the peristaltic pump aimed to reduce remaining control error, detailed description of this process is presented in the next chapter. *Control Selection Agent* selects the most suitable *CV* based on three factors. First factor is the prediction of expected result that will be observed, if offered *CV* is applied. This information is expressed as DO_{diff} - a distance between the expected extreme DO value and the set boundary (7.3). Next factor is agent's *reputation* which is assigned by *Diagnostics Layer* through an evaluation process. Offers from agents with higher *reputation* are preferred over ones from agents that are considered unreliable. The last factor is information about the pump used for control in *control offer*. Aeration should be mainly performed by the ON-OFF pump. Therefore, in a situation where multiple *control offers* from agents with reliable *reputation* declare satisfactory DO_{diff} values, the offer that uses the ON-OFF pump will be selected. Lastly, *Emergency Control Agent* is responsible to take over control when emergency occurs i.e. DO concentration drops below an emergency minimum or rises above an emergency maximum by turning on the ON-OFF pump or turning off all pumps respectfully. Emergency minimum is set by default at 1 (mgO_2/l) and emergency maximum at 3 (mgO_2/l).
- *Actuator Layer* – contains a single agent that is responsible for adapting control values prepared by the MAS into form used by the physical ON-OFF. In this case it transforms ON control value into series of 30 (s) cycles during which the aeration pump is turned on for 2 (s) – similar to *Control Cycle Agent* from chapter 4.
- *Diagnostics Layer* – is presented in detail in fig. 7.5 and consists of two agents. *Diagnostics Agent* uses knowledge about dependencies between *process values* and *system values* to detect faults in the system, diagnostics method is described in chapter 7.6. *Evaluation Agent* can assign three possible *reputation* grades: *reliable*, *uncertain* and *unreliable* to controller agents depending on their performance, evaluation process is described in chapter 7.7.

- *Interface Layer* – contains a single agent that gathers and presents to the user data about the controlled process and exposes interface through which user can interact with the system by e.g. changing a set point, manually initiating a diagnostic process etc..

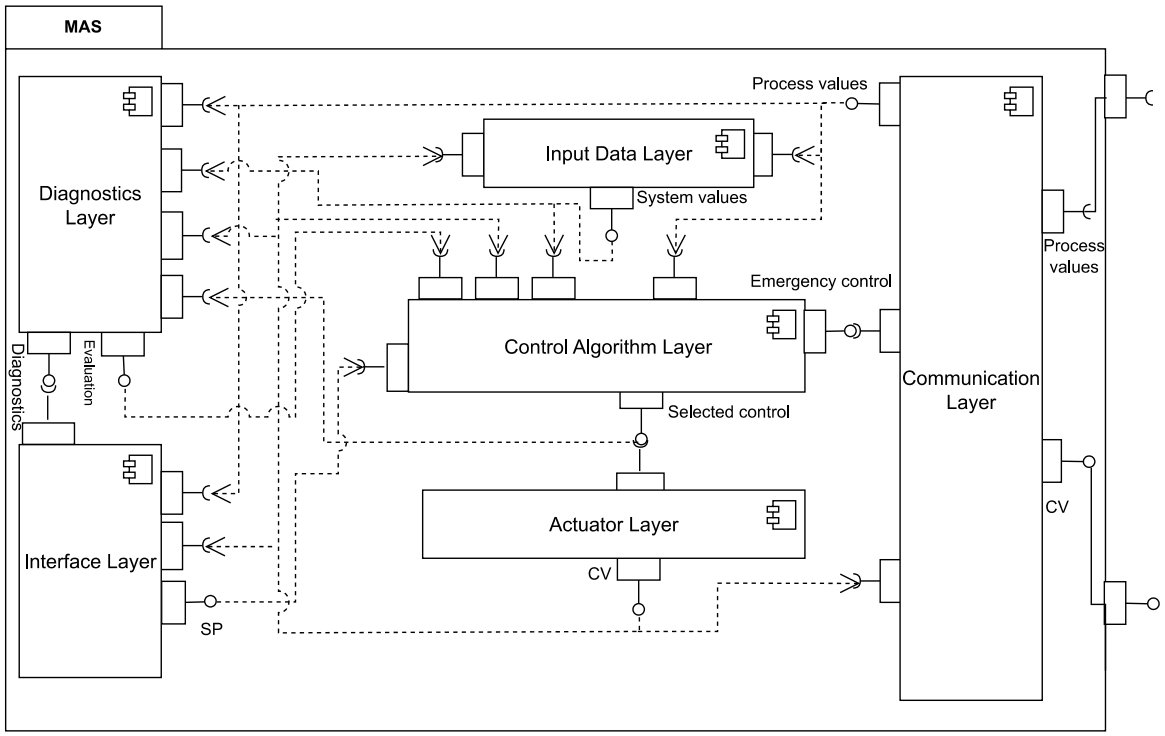

Fig. 7.2 MAS layered view

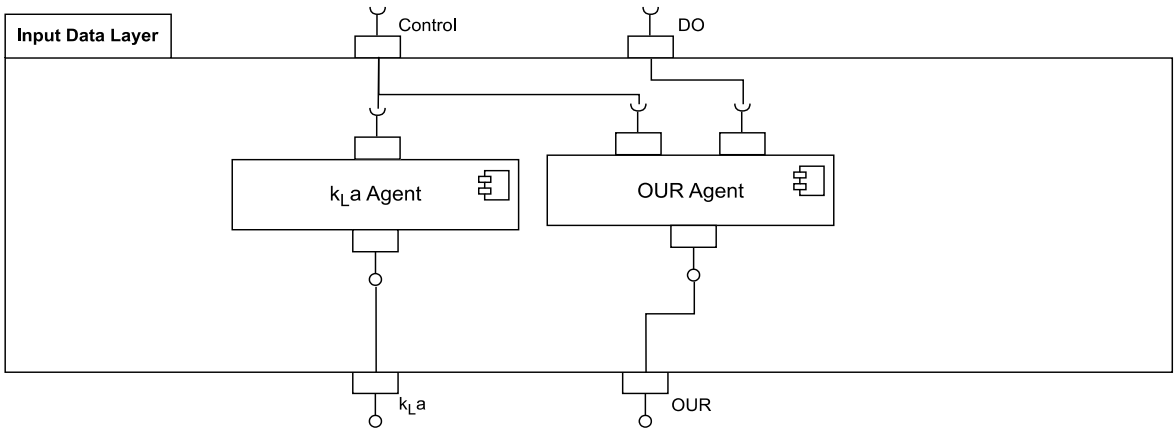

Fig. 7.3 Input data layer layout

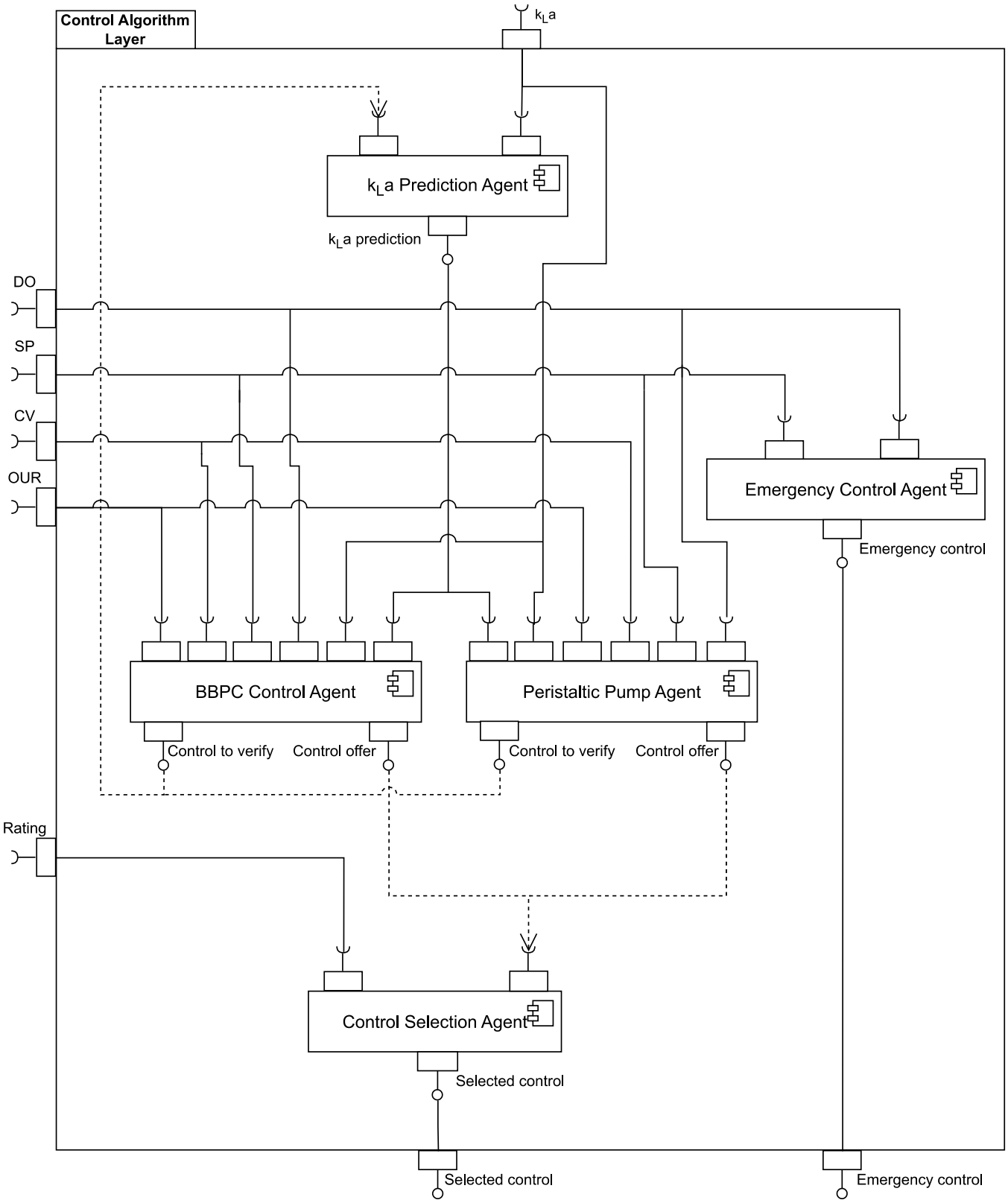

Fig. 7.4 Control algorithm layer layout

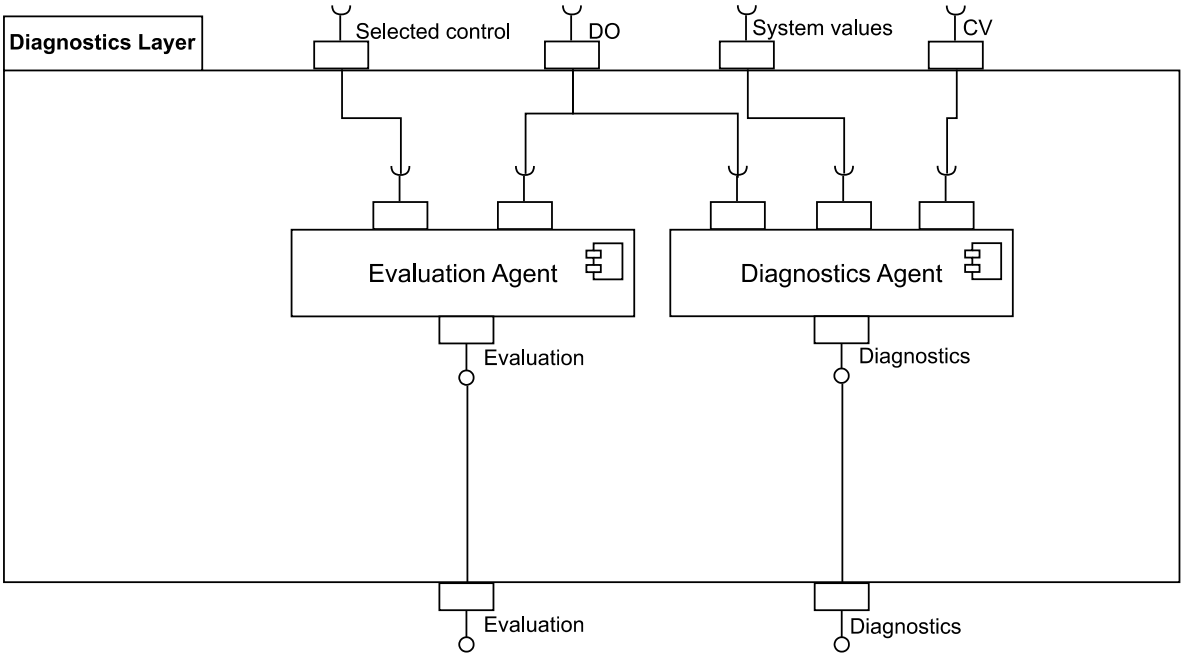

Fig. 7.5 Diagnostics layer layout

7.3. Ontology for presented MAS implementation

Ontology used alongside presented MAS implementation, presented in fig. 7.6 and fig. 7.7 is an extended version the ontology presented in previous chapter. Main additions are:

- Addition of concepts to *Diagnostics domain* that are used by *Diagnostics Agent* in process described in chapter 7.6.
- *Rate* action and *Reliability* property for *Agent* – these new addictions are dedicated for evaluation mechanism realized by *Evaluation Agent* that is described in detail in chapter 7.7.
- Remaining, smaller modifications include additions and editions of properties and are a specialization of general ontology schema to the specific control task.

Base ontology provided an efficient foundation for building a knowledge base for the multiple-input aeration system for the wastewater treatment process, resulting in significant saving of time and effort, which means that it fulfilled its objective.

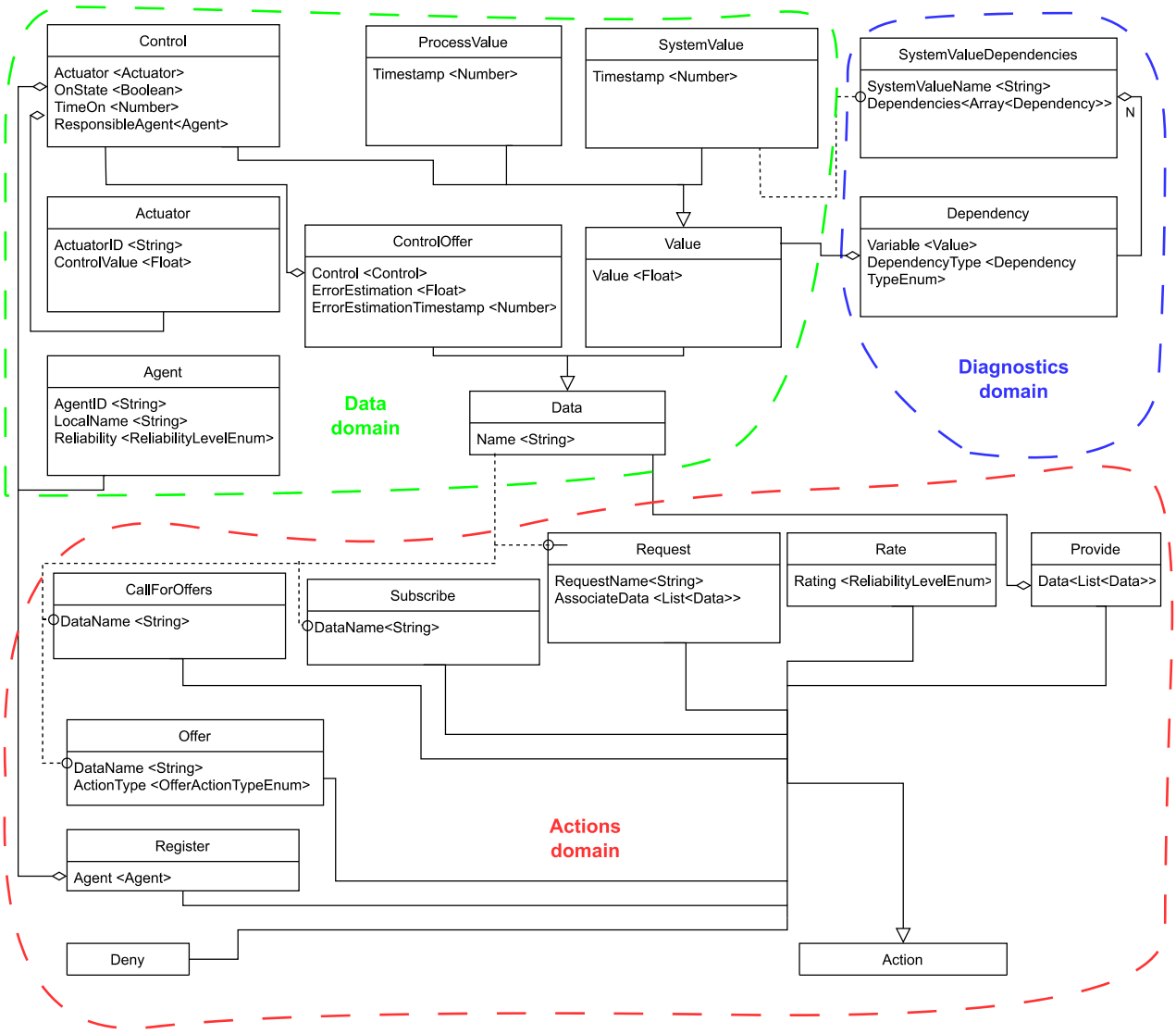

Fig. 7.6 Prepared ontology schema

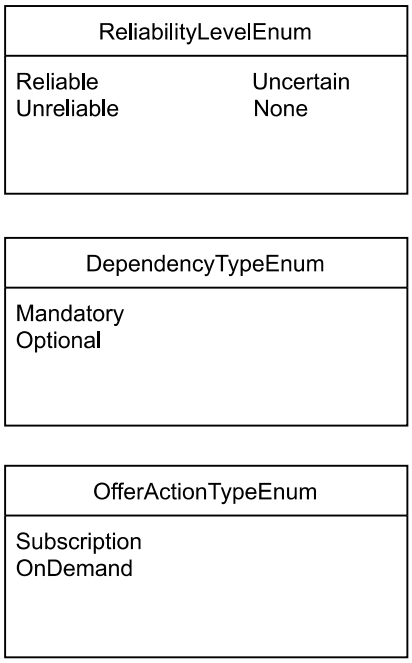

Fig. 7.7 Ontology additional types

7.4. Peristaltic pump control

Predictive control of aeration process using a peristaltic pump requires knowledge about pump's characteristics. To achieve that, u_{air} generated by pump over 30(s) cycle was estimated based on DO changes for multiple pump's settings. These settings include two parameters; one is pump's speed in 0-100 range where 0 means that the pump is turned off and 100 represents pump's maximum rotating speed. Second pump's parameter is running time (uptime) which describes how long the device was turned on in the 30(s) cycle. Obtained measurements allowed to prepare pump's characteristics in the form of an equation that is dependent only on these two parameters. However, these parameters occur in various combinations, thus resulting characteristics equation consists of many coefficients. Because of that, to maintain readability, the characteristics equation is presented in a tabular form in table 7.1. Based on observations during identification experiments, an additional constraint was applied, that the minimal possible setting for the peristaltic pump is 5(s) uptime in 30(s) cycle with a power level 10. These are the minimal settings for which the pump provides stable air output. Below them the pump's behaviour varied over time making its unreliable for process control purposes.

During agent's initialization, *Peristaltic Pump Agent* decodes provided pump characteristics and calculates $u_{air\_min} = u_{5s\_10}$ and $u_{air\_max} = u_{30s\_100}$, these boundary values are then used throughout agent's lifetime to look for suitable control values. Next, during each control cycle, a procedure of finding suitable control settings for peristaltic pump is performed. Its general view is presented in fig. 7.8. Figures 7.9 and 7.10 present how the prepared model estimation fits to measured data:

- The agent starts with requesting all process and system values, that are used as initial conditions for an aeration model.
- After that, the agent searches for u_{air_opt} that minimizes control error using a bisection method. Search starts at u_{air} equal to the average of u_{air_min} and u_{air_max}

and ends when the predicted extreme DO concentration is less than 0.005(mgO_2/l) from the set boundary or when the next step size is below $1 \cdot 10^{-6}$. In every step, the agent sends new u_{air_i} to k_La *Prediction Agent* that provides the estimated k_La response. This response is used by the agent to simulate the future behavior of the aeration process to predict the final control error. Depending on the control error values, new u_{air_i} is selected or process is stopped.

- Newly found u_{air_opt} needs then to be translated to peristaltic pump settings. This is a two-stage process. Firstly, the agent finds pump's maximum uptime duration τ(s) for which u_{air_opt} fits between u_{τ_10} and u_{τ_100}, search starts from 30(s) uptime value that gradually drops each step by 5(s). When uptime is detected, the agent moves to the second step. In this step, the agent already has the knowledge about u_{air_opt} and pump's uptime. Substituting these values into peristaltic pump's characteristics results in the quadratic equation that the agent solves to calculate pump's power value.

After the procedure is concluded, the agent saves detected pump's settings alongside the predicted control error to use it as a control for future *Control Selection Agent's* call for offers.

Table 7.1 Peristaltic pump model coefficients

		Power (P^X)		
		0	1	2
Uptime (τ^Y)	0	$4.92 * 10^{-2}$	$1.1 * 10^{-3}$	$1.24 * 10^{-5}$
	1	$3.1 * 10^{-3}$	$9.34 * 10^{-4}$	$-2.67 * 10^{-6}$
	2	$8.4 * 10^{-4}$	$4.91 * 10^{-5}$	$7.1 * 10^{-8}$
	3	$-1.02 * 10^{-4}$	$7.95 * 10^{-7}$	0
	4	$2.04 * 10^{-6}$	0	0

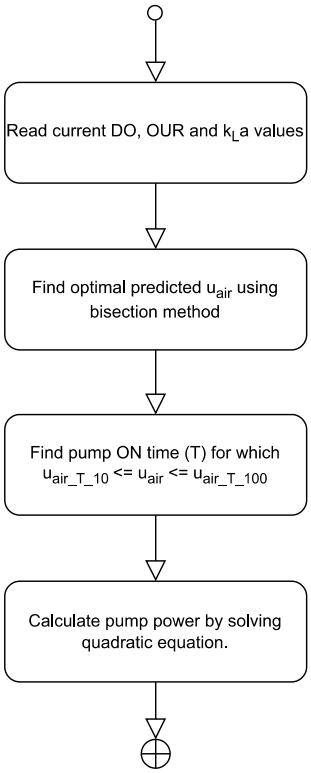

Fig. 7.8 Peristaltic pump settings deduction process

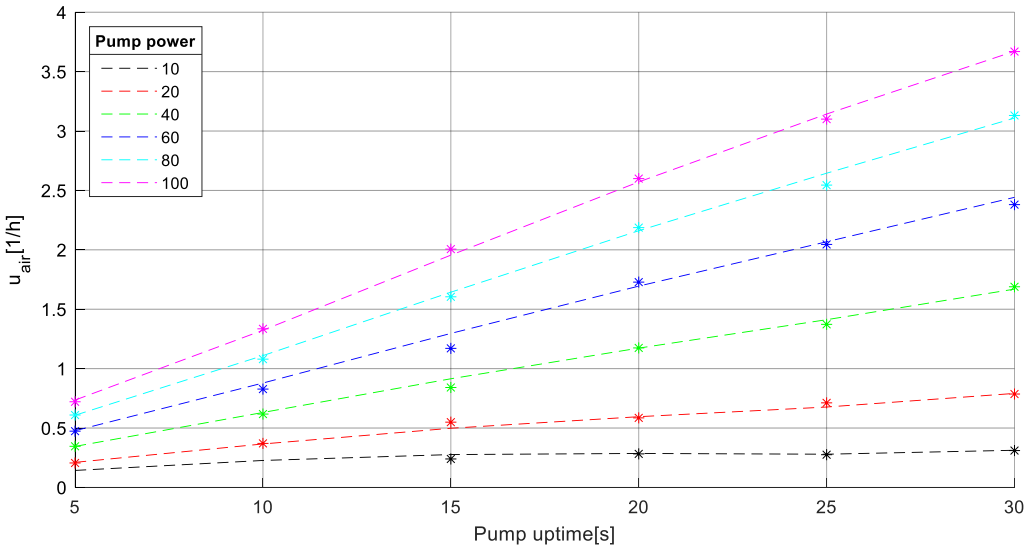

Fig. 7.9 Model's fitting over pump's uptime

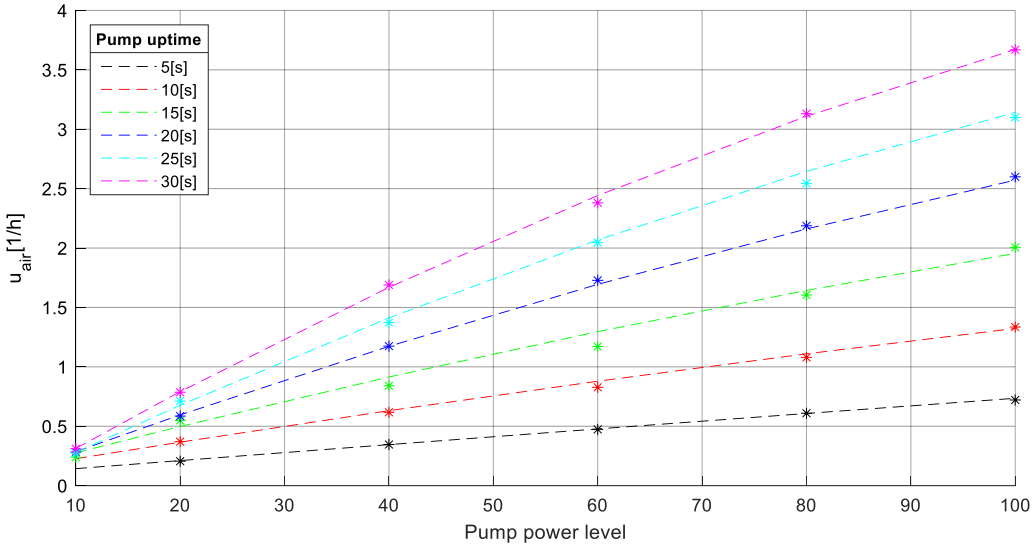

Fig. 7.10 Model's fitting over pump's power level

7.5. Fault detection procedure

Diagnostics Agent uses knowledge about dependencies between *process values* and *system values* to detect faults in the system. These dependencies are represented in the ontology (fig. 7.6 and 7.7). Every *system value* may have one or more dependencies. Diagnostic process is started manually or by agents, when an unexpected behavior is detected e.g. an agent requested a new value of some *system value* but did not receive any response. When that happens, the agent sends a request to the *Diagnostics Agent* to start diagnostics on a potentially faulty value. *Diagnostics Agent* begins the diagnostic process by sending a request to provide the potentially faulty value. From that point, one of three possible scenarios might occur:

- Positive reply when the requested value is received. Such reply indicates that the system works correctly, thus no fault is detected.
- The Diagnostic Subsystem may also receive a negative reply which indicates that agents responsible for providing the requested value work correctly but due to outside factors are unable to provide the value. In such a case, the Diagnostic Subsystem uses its knowledge base to get dependencies of the value and starts diagnostic processes for these dependencies.
- The last scenario assumes that no reply is received in a specified time, which indicates that a fault occurred in agents responsible for providing the requested value.

The result of the diagnostic process is presented in the user interface. This process in the form of decision diagram is presented in fig. 7.11.

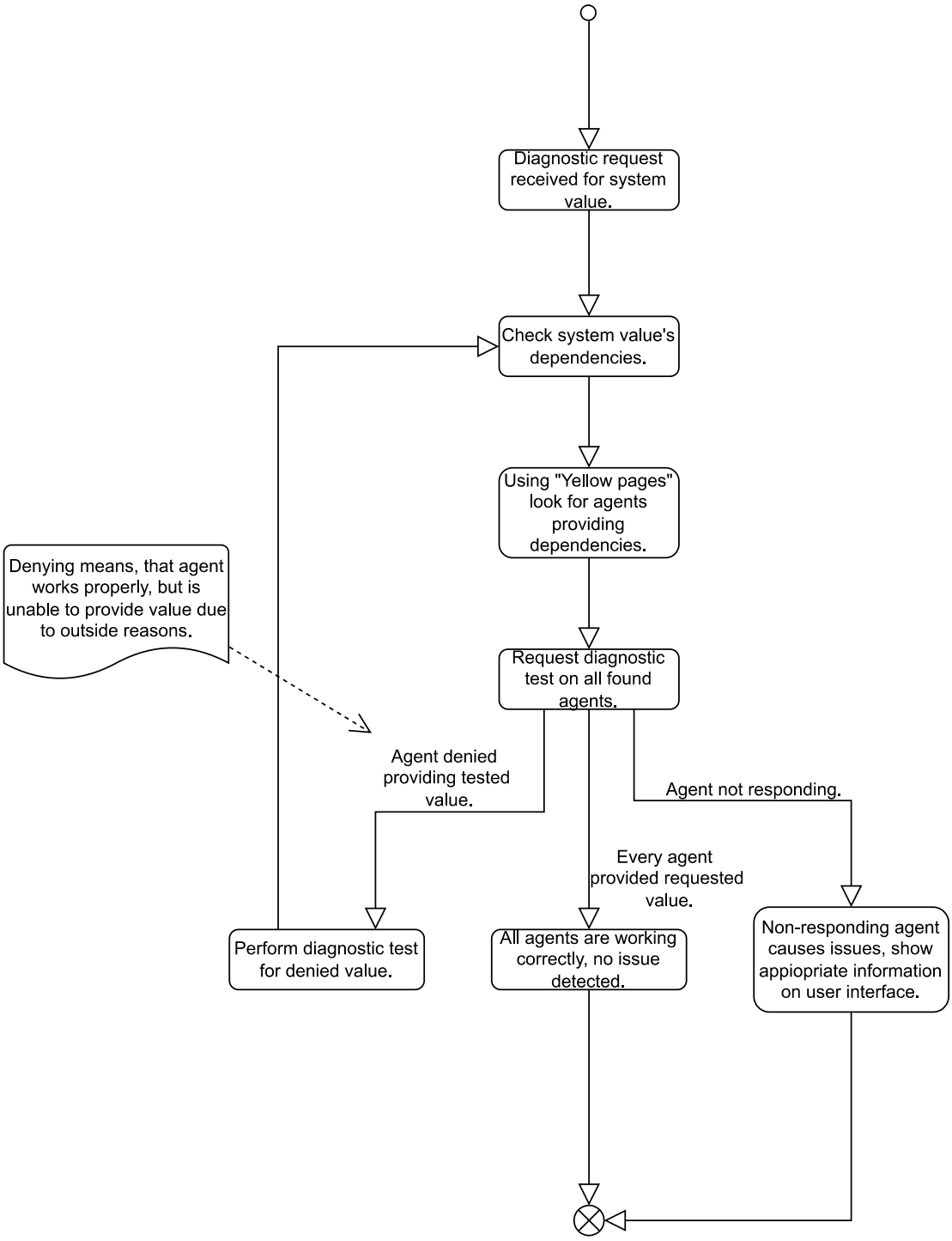

Fig. 7.11 Graphical representation of diagnostics process

7.6. Reliability rating system

Evaluation Agent can assign three possible *reputation* grades: *reliable*, *uncertain* and *unreliable* in a way presented in fig. 7.12. Grade *reliable* indicates that other agents should trust in declarations provided by the agent with this grade. On the other hand declarations from agents with grade *unreliable* should not be trusted. *Uncertain* is a transitional grade which indicates that agent's reputation may soon change to another grade. Agents with this grade might be trusted by others, but it is not mandatory. Agents

that require high reliability of input data should rely only on *reliable* agents. Reputation grades are assigned based on a verification of expected control result that the agent declared in its *control offer*. If agent's declaration matches the observed result, its reputation is raised by one grade until it reaches *reliable* grade, otherwise its reputation is dropped by the grade until it reaches *unreliable* grade.

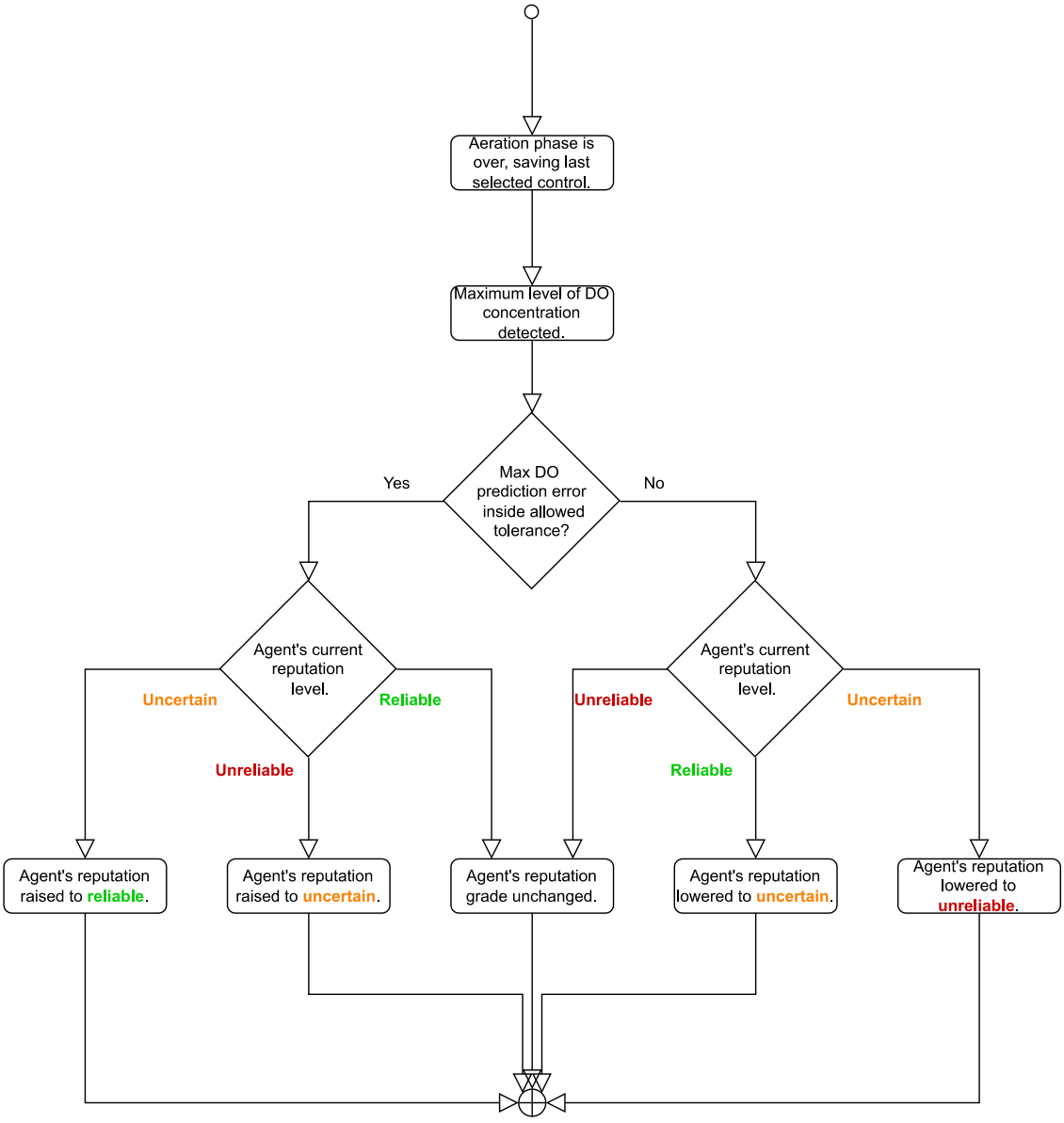

Fig. 7.12 Graphical representation of an evaluation flow

7.7. Simulation results

At the beginning, initial measurements to estimate u_A for the ON-OFF pump and u_{air} values range for the variable speed pump were performed using the approach described in [26]. For the ON-OFF pump u_A was estimated at 2.52 (1/h), the variable speed pump characteristic coefficients are presented in table 7.1. Simulation validation was performed using the presented multi-agent control system and LabVIEW environment where the

aeration process described with (3.1) was simulated using previously estimated u_{air} values. Other parameters were set as following: DO_{sat} = 10 (mg O_2/l), T_{kLa} = 88 (s), T_0 = 27 (s), *OUR(t)* was set at the constant value during the simulation equal to 2.4 (mg O_2/l h). Process simulation was using LabVIEW OPC UA Toolkit to communicate with created MAS.

Firstly, two control methods were analyzed and compared through the simulation. In the first approach, the simulated process was controlled by the conventional BBPC algorithm. For the next approach, control was performed by the MAS implementation described in chapter 7.2. Results of simulation are presented in fig. 7.13.

Obtained results show that for both approaches SP_{min} value was reached precisely, but the precision of reaching SP_{max} value varies between them. For BBPC, significant overshoots over SP_{max} boundary are visible (around 0.1 (mgO_2 / l)), this behavior is expected and matches results and analysis presented in [26]. MAS with the variable speed pump control can achieve significantly better precision of reaching SP_{max}. Differences between peaks of DO values and SP_{max} for the second approach were not larger than 0.006 (mgO_2 / l). *CV* values in fig. 7.13 illustrate how MAS control with the variable speed pump functions. Drop of the u_{air} from 2.52 (1/h) to the other non-zero value indicates the moment when MAS decides to turn on the variable speed pump tuned to precisely reach SP_{max} boundary. For standalone BBPC u_{air} value can only be switched between 0 (1/h) and 2.52 (1/h), because of that, during the last aeration cycle u_{air} must stay at 2.52 (1/h) or 0 (1/h) which usually results in either a noticeable overshoot or a maximum value not reaching SP_{max}.

During the next simulation, MAS reputation and reconfiguration mechanisms were verified. Simulation started using the MAS system with the variable speed pump, but parameters of this pump's model were intentionally set incorrectly. Later, agents with correct variable speed pump's model parameters were added to MAS. Results of the simulation presented in fig. 7.14 visualize influence of *Evaluation Agent* actions on the process control and MAS ability for online reconfigurations. For the first two aeration periods MAS uses the incorrectly modelled variable speed pump. This can be noticed by a visible difference between the maximum DO declaration and the real maximum DO value. Because of that, in each of these aeration periods, reputation of agents responsible for the variable speed pump control is dropped by one grade down to *unreliable* grade. After two aeration periods, control offers graded as *unreliable* from agents controlling the variable speed pump are ignored and as a result BBPC algorithm based only on the ON-OFF pump is realized. That results in visible gaps between the SP_{max} and the maximum DO value but expected maxima now matches measured values, proving that agents responsible for this control are *reliable* and their offers should be used over *unreliable* ones. The moment when new agents were added to MAS is indicated by the vertical line. Newly added agents were seamlessly included in MAS control flow and in the next aeration cycle their control offer was already in use resulting in immediate improvement in precision of reaching SP_{max} value. This proves that the MAS has an ability to adopt new agents to its structure during runtime. Therefore, it is possible to provide online improvements and corrections to MAS and perform its reconfigurations.

Lastly, the MAS ability to diagnose its faults was verified. For this experiment, the agent responsible for *OUR(t)* calculations was modified to impair its communication capabilities and thus become a fault point of the MAS. When the control system was started, the agent responsible for diagnostic capabilities was requested to diagnose the state of *CV* calculation. Every event that might indicate a fault in the system is presented by the agent on its GUI component. Such events occurred during *CV* test, control-forward test (in the MAS implementation control-forward is the name of the selected control) and lastly in the *OUR(t)* test where nonresponding agent was detected. Example of UI communicates is presented in fig. 7.15.

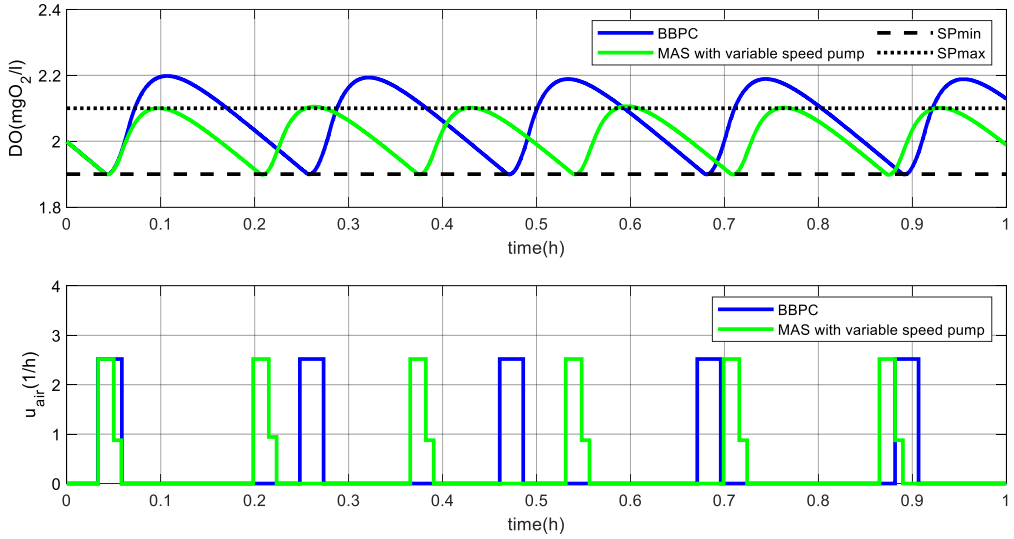

Fig. 7.13 Simulation results of DO concentration control using different control algorithms

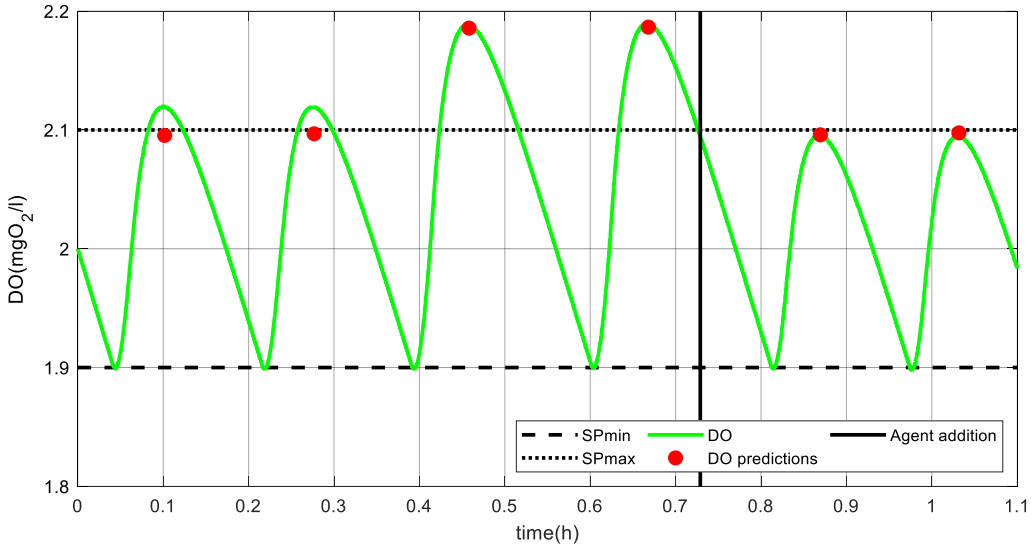

Fig. 7.14 Simulation verification of reputation mechanisms and MAS reconfiguration

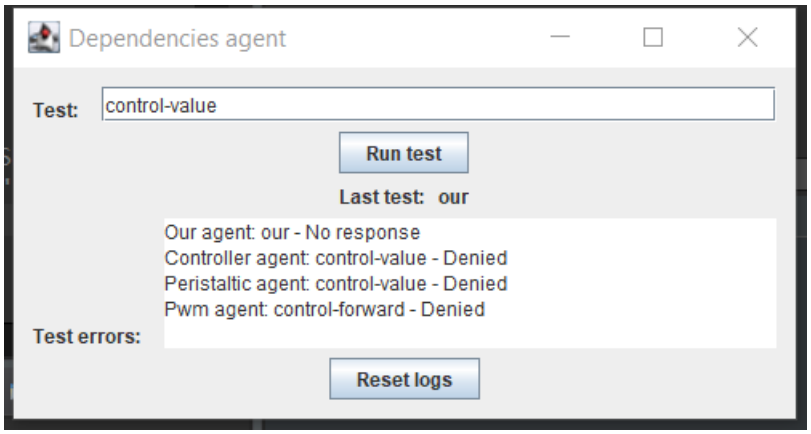

Fig. 7.15 GUI messages for the diagnostic experiment with detected fault

7.8. Experimental results

Data transfer between MAS and PC with SCADA deployed is performed using OPC UA protocol as presented on fig. 7.16. Experimental validation was made using the real activated sludge laboratory setup presented in chapter 3 with preliminary off-line estimation of parameters of the model based on measurement data. Assuming constant saturation concentration $DO_{sat} = 10$ (mg O_2/l), parameters values were estimated as following: $T_{kLa} = 50$ (s), $T_0 = 14$ (s) and $u_A = 2.52$ (1/h). First experiment was focused on comparison of the presented multi-agent control system with BBPC algorithm. During experimental measurements *OUR(t)* value was estimated at *OUR(t)* = 2.7 (mgO_2/l h).

As can be seen in fig. 7.17, overshoots generated by the BBPC algorithm are visibly smaller than ones obtained through simulation. Even though, in comparison to simulation results, overshoots during BBPC control for these conditions are significantly smaller, the proposed agent-based solution is still able to outperform the former controller and reach SP_{max} boundary precisely during every aeration cycle by properly adjusting *CV* value for the last aeration cycle in the sequence. *CV* graph in fig. 7.17 shows that for some aeration cycles there was no need to use the variable speed pump to achieve high control precision. MAS recognized these cases and used only the ON-OFF pump.

Because it is not possible to prepare an organic substrate batch that would ensure identical *OUR(t)* for each experiment, slight differences in *OUR(t)* values can be noticed through descending *DO* slopes not being parallel to each other in fig. 7.17. However, in author's opinion influence of those differences on final results are negligible.

The goal of the second experiment was to verify the implemented MAS response to addition and removal of new controllers. The experiment was divided into three phases. Firstly, agents responsible for the variable speed pump control were absent, forcing the control system to use only agent-based version of BBPC algorithm. After some time, agents that are capable to control the variable speed pump were added to the system. Then after 1.5(h), those agents were removed, forcing the system to once again rely only on the BBPC algorithm.

Results in fig. 7.18 show that the control system can switch to more precise control the moment it becomes available. When capabilities of the control system are limited by removing specific agents, the MAS is also able to immediately switch to a less precise control without any disruptions.

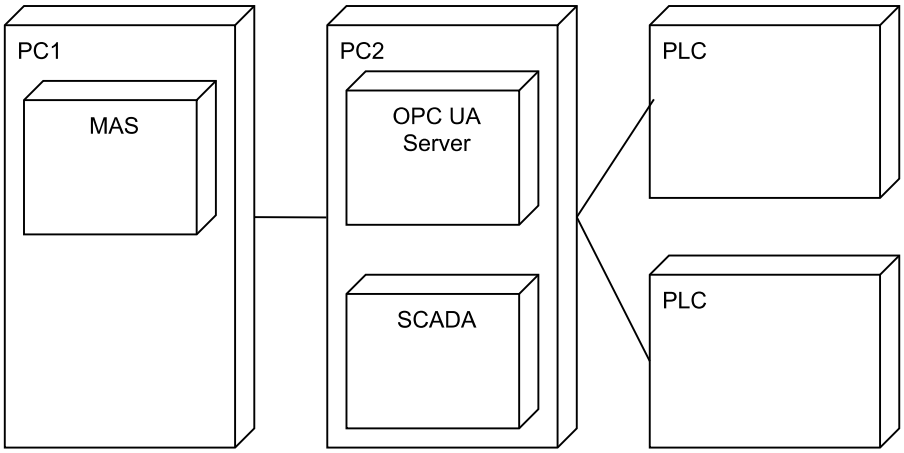

Fig. 7.16 Deployment diagram of example implementation

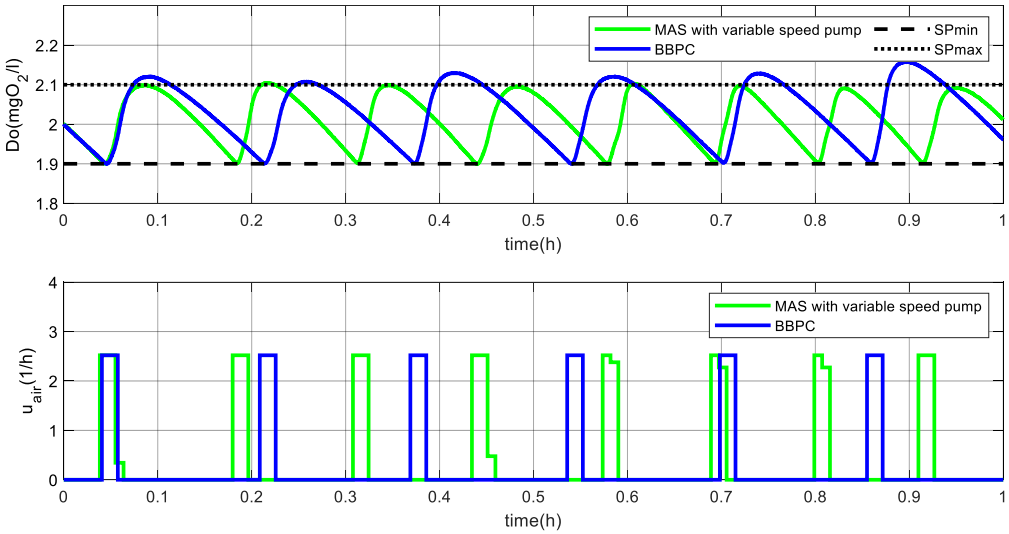

Fig. 7.17 DO concentration control for OUR = 2.7 (mg O2/lh) and corresponding CV values

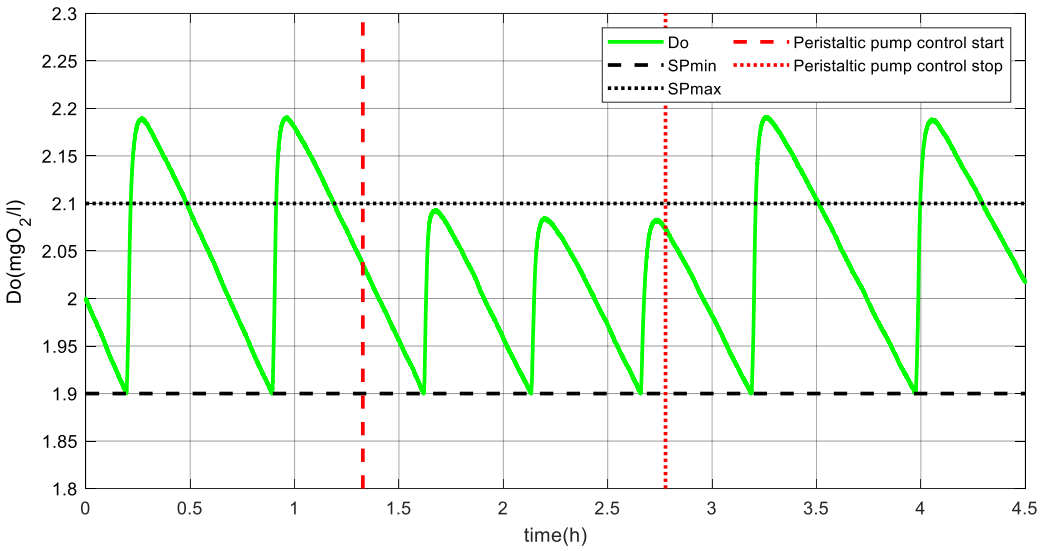

Fig. 7.18 DO concentration during addition and removal of supplementary controller agent

7.9. Summary

This chapter presented the concept of a multi-agent, multi-controller system for a model-based predictive control of continuous processes.

The proposed MAS design was verified in a practical implementation of a boundary-based control of a DO concentration in a biological reactor aerated by an ON-OFF and a variable speed aeration pumps. For that implementation a novel, agent-based control algorithm was proposed and tested in both a simulation environment as well as in a real activated sludge laboratory setup. Practical comparison of the novel control system with BBPC algorithm proved superiority of MAS that was able to reach set boundaries, especially SP_{max} boundary with a significantly better precision. Further experiments additionally verified, that parts of presented MAS can be added or removed from the system at runtime with seamless switch in control. MAS is also able to reconfigure its structure and switch between available control algorithms based on reputation mechanism that compares agents' declarations with real measurements and assigns reputation grades based on that knowledge. Additionally, a fault detection algorithm that uses knowledge about the system to detect faulty agents is proposed and verified. These attributes improve MAS maintainability, using such MAS based control systems might reduce downtime and costs spent for maintenance in industrial use cases.

Main conclusions from this chapter in the context of the dissertation is the fact that MAS prepared according to the proposed ontology structure and the architecture layout was successfully applied in control of the complex biotechnological process. MAS architecture layout proved its usability and the previously prepared base implementation of ontology significantly reduced time needed for preparation of system's knowledge base. MAS by itself was able to effectively control the aeration process outperforming a complex, conventional control algorithm, and display self-diagnostic and reconfigurability behaviors. This proves that with enough knowledge and preparation MAS based solution might be a competitive alternative to conventionally built systems. Suggested design of MAS for a model-based predictive control of continuous processes was successfully applied to control of a real continuous process using multiple actuators and various types of control laws. However, up to this chapter all of the work was built for and verified on the aeration process. Therefore, the approach presented in this chapter needs to be additionally verified on other types of biotechnological processes.

8. MAS for continuous control of lactic acid fermentation process

This chapter describes the last phase of the doctoral research that focuses on an implementation of the previously prepared MAS based approach for the continuous control of process other than a dissolved oxygen concentration control in wastewater treatment.

As mentioned in the previous chapter - up to this phase, all of the work was built for and verified only using the aeration process. To ensure reliability of the proposed approach, it needs to be additionally verified using a different type of a biotechnological process. For that task, control of a lactic acid fermentation was chosen.

This chapter focuses on an application of MAS developed according to prepared rules for the lactic acid fermentation control in a two tank reactor setup described in chapter 3 of the dissertation. Similarly to previous research – prepared MAS should be able to reconfigure itself according to encountered conditions through the seamless switch between control algorithms.

8.1. Definition of control objective and used controllers

The universal objective in the lactic acid fermentation process control is to optimize conversion of glucose into lactic acid via fermentation. According to analysis from [36], this can be indirectly achieved by keeping substrate concentration S, around the specific point, that ensures maximum efficiency of lactic acid production, while minimizing residual glucose concentration at the same time. This approach is applied to the control of the simulated lactic acid fermentation process described in chapter 3, as a result – S_1 and S_2 become output values of the process that should be kept at a specified level while substrate input dilution rates D_1 and D_2 are control values. However, it is assumed that the enrichment factor dilution rate (D_{12}) is constant throughout the experiment, resulting in final pair of control values being D_{11} and D_2. Additional assumptions are as follows:

- specific growth rates $\mu_{1,}$ $\mu_{2,}$ υ_1, $\upsilon_{2,}$ q_1 and q_2 are not known,
- measurements of biomass and lactic acid concentrations for each reactor are not available,
- specific values of inflow substrate concentrations S_{in1}, and S_{in2}, are unknown, however ranges of possible concentrations $<S^-_{in1}, S^+_{in1}>$ and $<S^-_{in2}, S^+_{in2}>$ are available,
- the remaining projects coefficients, e.g. enrichment factor inflow concentration α_{in1} are known.

Two controller types were chosen for this task. One is a conventional PI controller that will be used as a backup controller. The next control algorithm used is a Robust Adaptive Controller – RAC presented in [37] which will be used as a main controller for the experiments.

8.1.1. RAC Controller

RAC Controller, described in [37] is an adaptive control strategy dedicated to the lactic fermentation process control and designed to answer the process's main control

challenges, i.e. its strong nonlinearities and uncertainties. Algorithm's control law is presented on equations (8.1-8.5).

$$\dot{\widehat{S_1}} = -\widehat{\rho_1} + \frac{D_{11}(S_{in1}^{-}+S_{in1}^{+})}{2} - D_1 S_1 + \omega_1(\widehat{S_1} - S_1) \quad (8.1)$$

$$\dot{\widehat{S_2}} = -\widehat{\rho_2} + \frac{D_2(S_{in2}^{-}+S_{in2}^{+})}{2} + D_1 S_1 - (D_1 + D_2)S_2 + \omega_2(\widehat{S_2} - S_2) \quad (8.2)$$

$$\dot{\widehat{\rho_1}} = \gamma_1(\widehat{S_1} - S_1) \quad (8.3)$$

$$\dot{\widehat{\rho_2}} = \gamma_2(\widehat{S_2} - S_2) \quad (8.4)$$

$$\begin{bmatrix} D_1 \\ D_2 \end{bmatrix} = \begin{bmatrix} \frac{S_{in1}^{-}+S_{in1}^{+}}{2} - S_1 & 0 \\ S_1 - S_2 & \frac{S_{in2}^{-}+S_{in2}^{+}}{2} - S_2 \end{bmatrix}^{-1} \cdot \left\{ \begin{bmatrix} \dot{S_{SP1}} \\ \dot{S_{SP2}} \end{bmatrix} + \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \cdot \begin{bmatrix} S_{SP1} - S_1 \\ S_{SP2} - S_2 \end{bmatrix} + \begin{bmatrix} \widehat{\rho_1} \\ \widehat{\rho_2} \end{bmatrix} + \begin{bmatrix} \frac{D_{12}(S_{in1}^{-}+S_{in1}^{+})}{2} \\ 0 \end{bmatrix} \right\} \quad (8.5)$$

Where $\widehat{\rho_1}$ and $\widehat{\rho_2}$ are estimations of reaction rates inside each reactor, γ_1, γ_2, ω_1, and ω_2 are estimator's tuning parameters and λ_1, λ_2 are controller's gains. For this experiment, values for each parameter were based on analysis from [37]: $\gamma_1 = 0.5, \gamma_2 = 0.4, \omega_1 = -0.75, \omega_2 = -0.5, \lambda_1 = \lambda_2 = 0.25$.

8.1.2. PI Controllers design

It is often challenging for PI controllers to be able to follow a set point for such complex, nonlinear processes like lactic fermentation. To ensure satisfactory control quality, process's characteristics need to be analyzed. In this case static and dynamic characteristics of each reactor were analyzed, during analysis of the second reactor assumption was made, that output substrate concentration and dilution rate from the first reactor were constant and equal to 2(g/l) and 0.0556(1/h) respectively. For the reactor 1, input substrate concentration S_{in1} during analysis was kept at 50(g/l) and input enrichment factor inflow concentration α_{in1} at 6(g/l), for reactor 2 S_{in2} was set at 200(g/l). Results of process analysis are presented in fig. 8.1-8.4, figures 8.1 and 8.3 show nonlinear characters of reaction inside each reactor. Because of that, PI controllers tunings will be efficient only for specific set point values. It was decided, that PI controller for the reactor 1 will be tuned for S_1=2(g/l) and the reactor 2 will be tuned for S_2=5(g/l). Tuning for these values were derived using fragments of step responses from figures 8.2 and 8.4 that correspond to chosen values. Based on these step responses, FOPDT models were identified that were finally used to calculate PI tunings, presented in (8.6, 8.7).

$$PI_1(s) = 0.25\left(1 + \frac{1}{1.97s}\right) \quad (8.6)$$

$$PI_2(s) = 0.015\left(1 + \frac{1}{1.88s}\right) \quad (8.7)$$

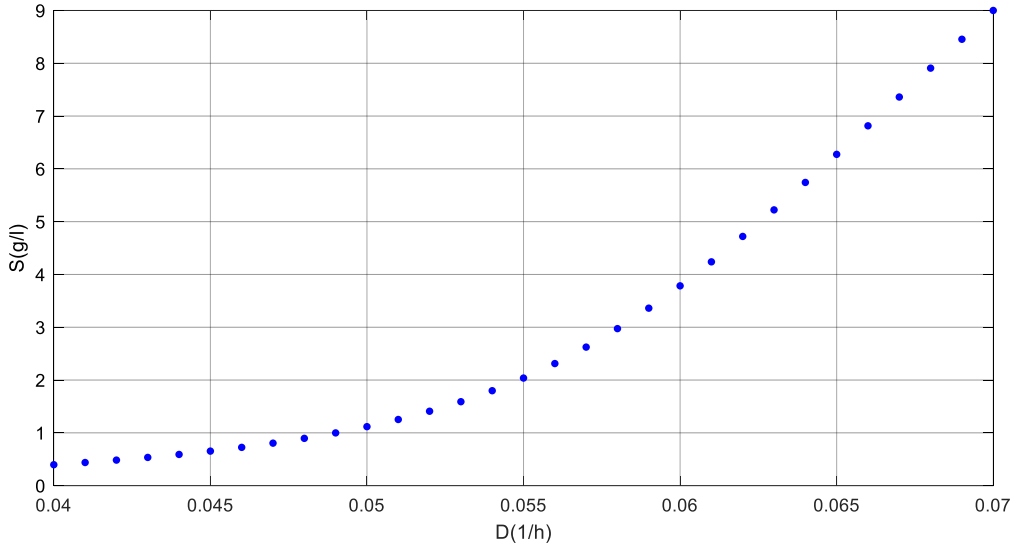

Fig. 8.1 Static characteristics of S_1 over D_{11}

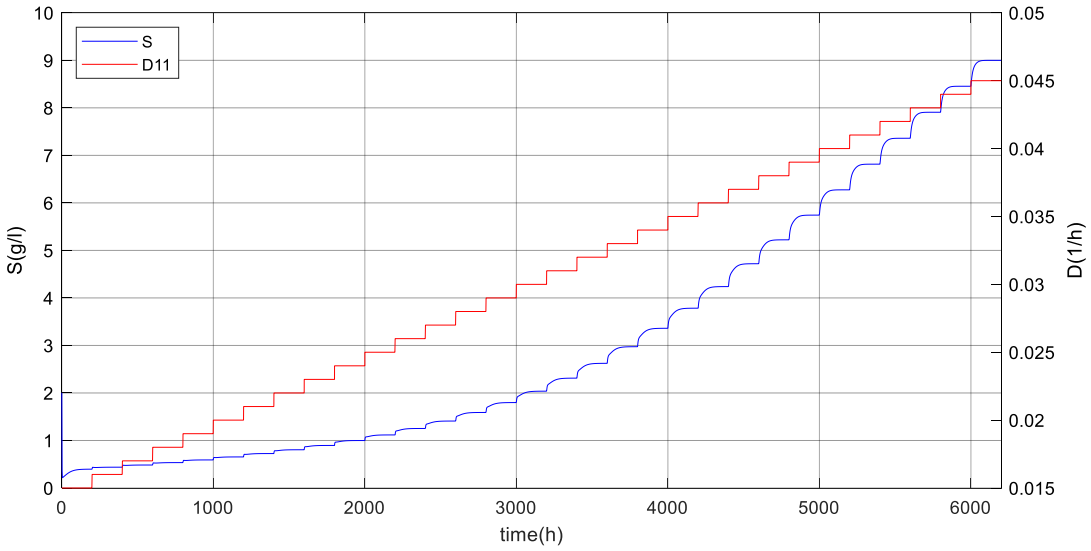

Fig. 8.2 S_1 changes over time during static characteristics analysis

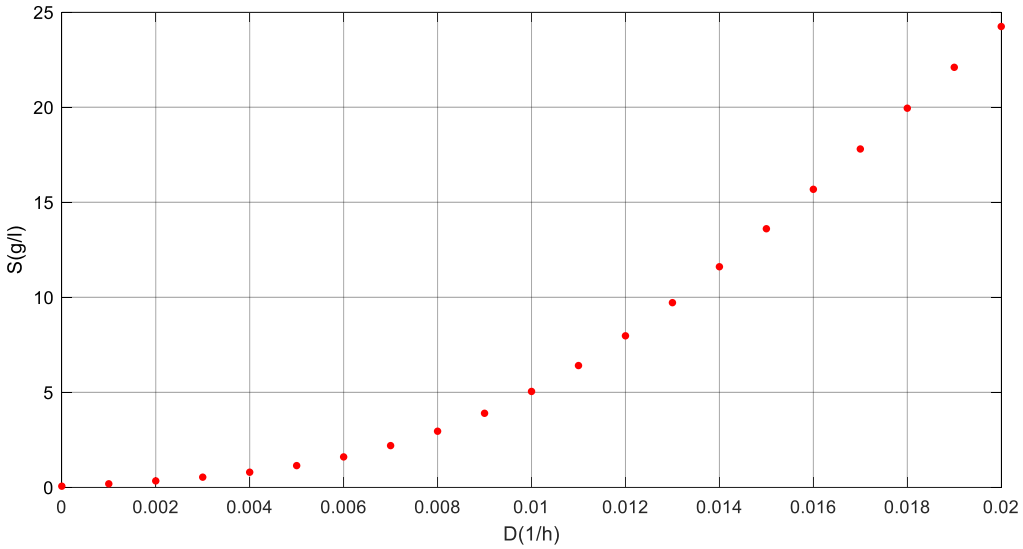

Fig. 8.3 Static characteristics of S_2 over D_2

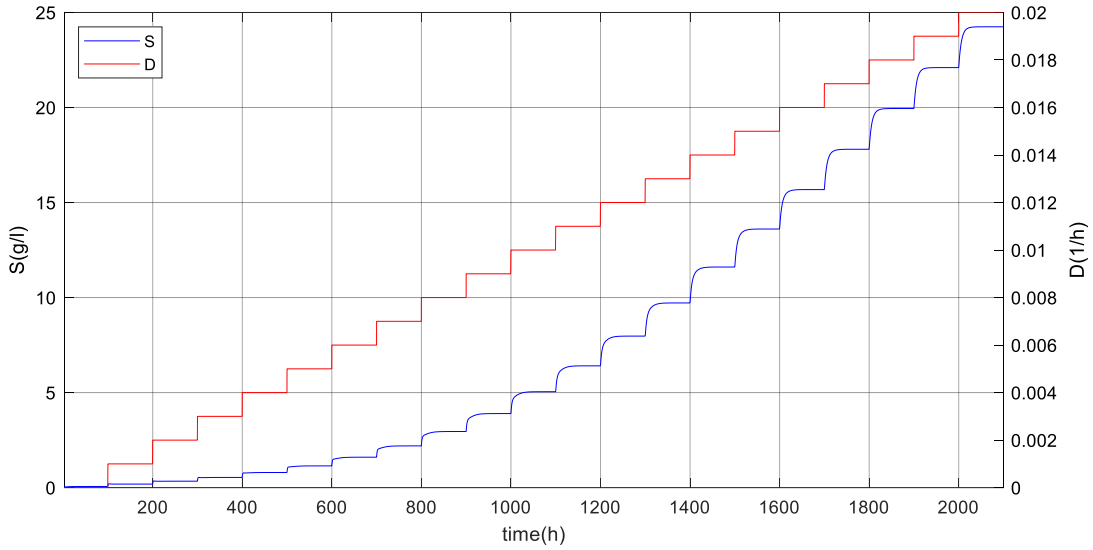

Fig. 8.4 S2 changes over time during static characteristics analysis

8.2. MAS implementation architecture

MAS architecture was designed based on prepared rules and is presented in fig. 8.5, its layer view is similar to the MAS architecture used in previous research presented in the last chapter. The only difference is a lack of diagnostics layer – this project is strictly focused on application of MAS to process control, thus diagnostic capabilities of MAS were in this case omitted. Every layer contains following agents:

- *Communication Layer* – once again contains single agent that operates OPC UA client. However in this case multiple *process values* are transferred to the MAS from the controlled system, these values are S_1,S_2,S^-_{in1}, S^+_{in1},S^-_{in2}, S^+_{in2}. This agent sends to OPC UA server *CV* values (D_{11} and D_2) that are prepared by the MAS.
- *Input Data Layer* –this layer consists of a single agent that derives reaction rate estimations for each reactor ($\widehat{\rho_1}$ and $\widehat{\rho_2}$). This agent is sending their calculated values on-demand – when an appropriate request is requested.
- *Control Algorithm Layer* – consists of five agents organized in a way presented in fig. 8.6. *RAC Control Agent* calculates control offers using RAC control law. *PI1 Agent* calculates control offers of D_{11} using PI controller with tunings from (8.6) and *PI2 Agent* analogically calculates control offers of D_2 using tunings from (8.7). There are two agents responsible for selection of the most suitable control values – one for each control value, their names are *D_{11} Selection Agent* and *D_2 Selection Agent*. In this case the selection is based on a priority mechanism – agent with higher priority will take over control whenever it is able to derive the control value. In presented experiment *RAC Control Agent* has the highest priority.
- *Actuator Layer* – contains single agent that packs data from *Control Algorithm Layer* into a single message format and propagates it throughout the system.
- *Interface Layer* – contains a single agent that gathers and presents to user data about the controlled process and exposes interface through which the user can modify control set points.

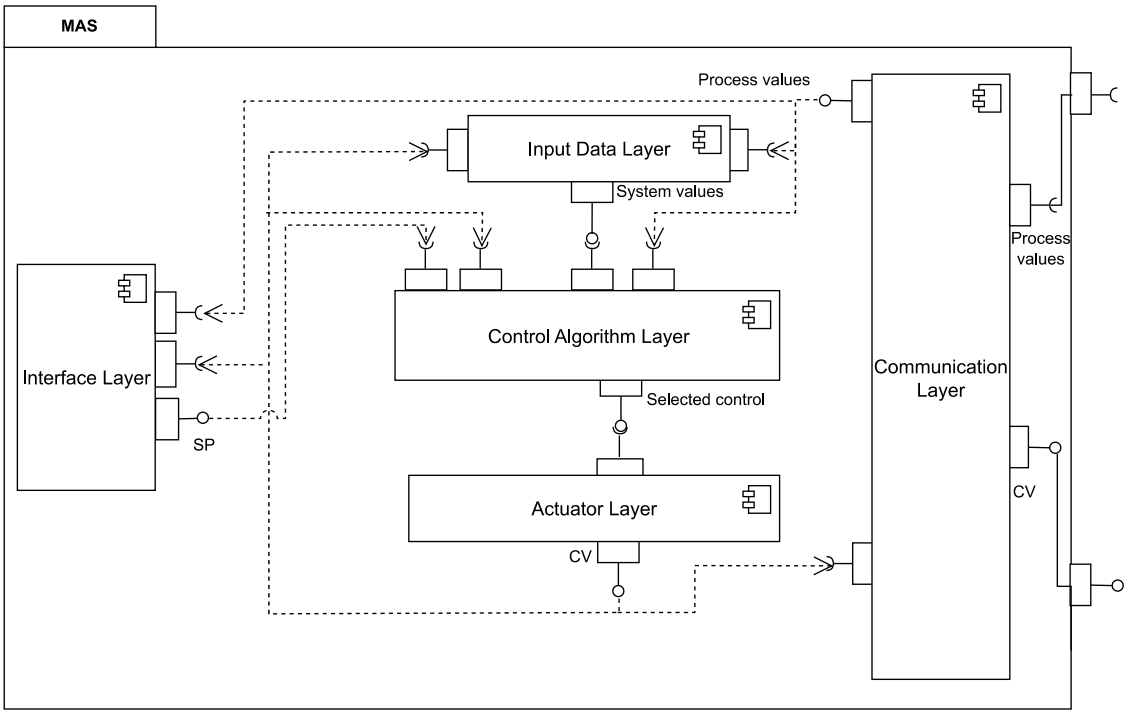

Fig. 8.5 MAS architecture layer view

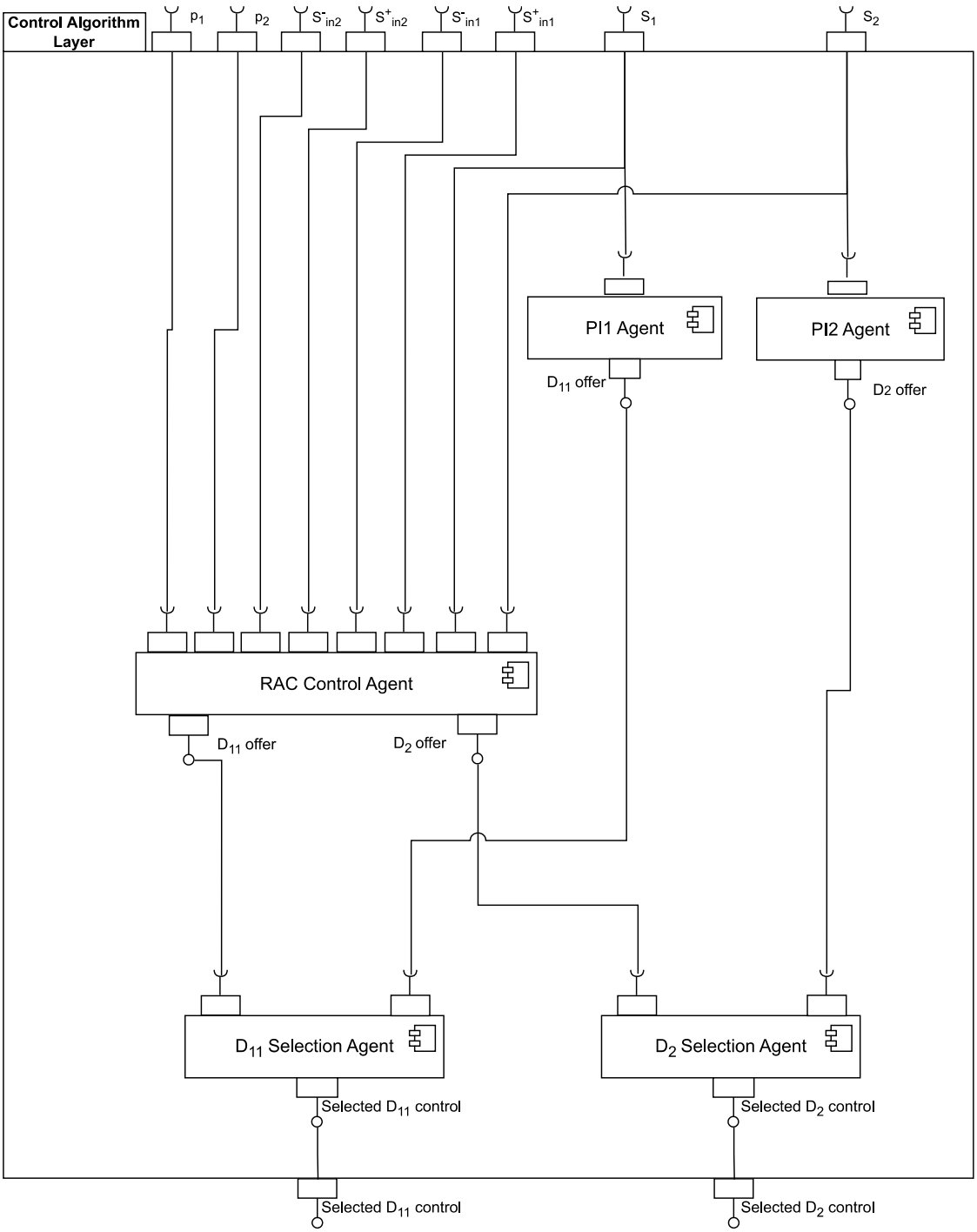

Fig. 8.6 Control algorithm layer view

8.3. Ontology in presented MAS implementation

Ontology used alongside presented MAS implementation, presented in fig. 8.7 is a reused version of the ontology described in the previous chapter with minor adjustments. These adjustments include removal of entities and attributes used in diagnostics and rating procedures, because they are not used in this project. Additionally, *ControlOffer* attributes were adjusted to reflect a different, priority based control selection algorithm. Such broad reuse of previously prepared ontology proves reusability of designed approach.

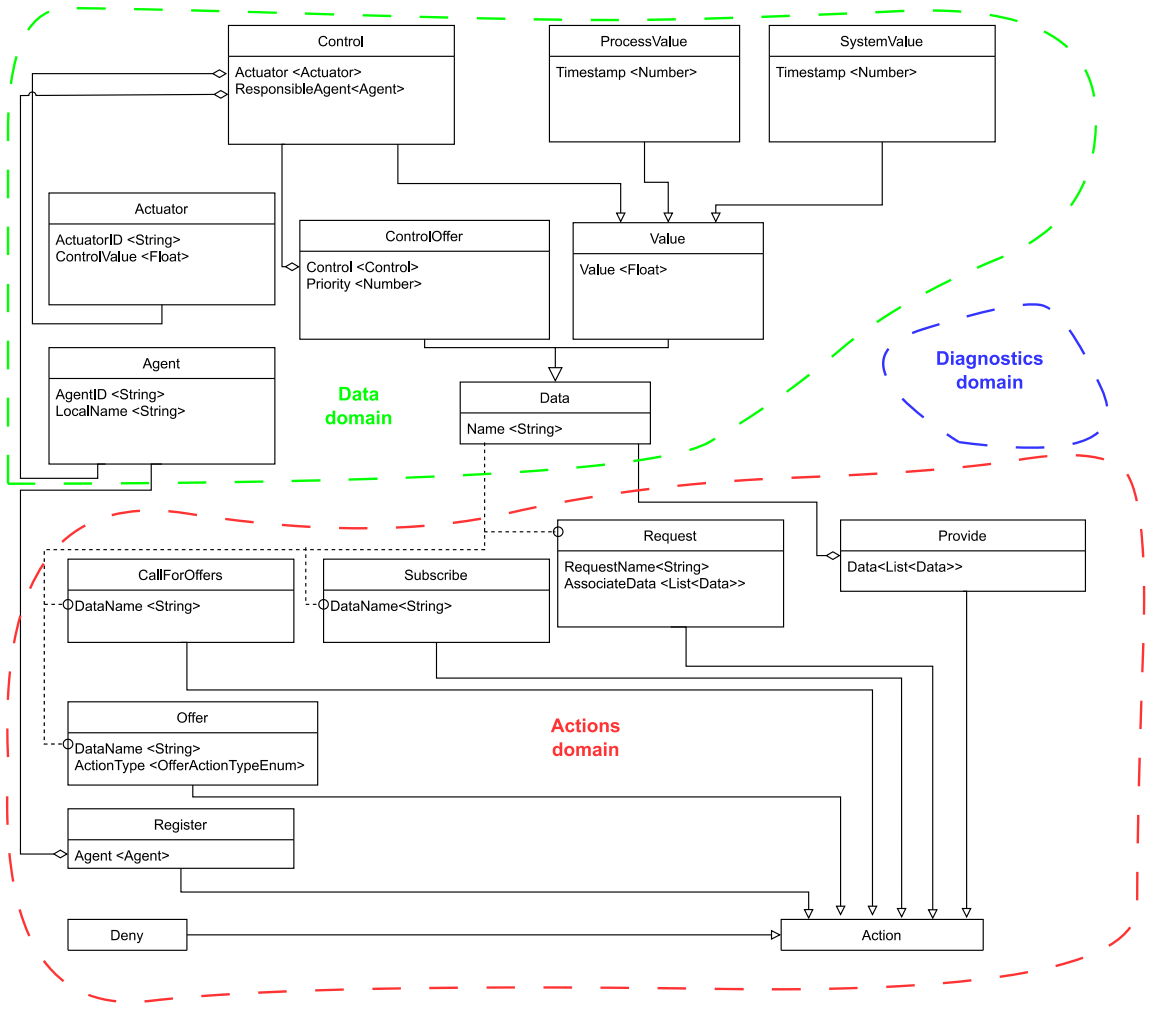

Fig. 8.7 Ontology used during the experiment

8.4. Experimental results

Experiment was performed on a setup presented in the deployment diagram from fig. 8.8. The central part of the setup is an OPC UA Server built in LabVIEW, remaining components connect to the server using their OPC UA Client implementation. MAS, described in the previous chapters is built using JADE framework, MATLAB S_{in} generator is responsible of providing S_{in1}, S^{-}_{in1}, S^{+}_{in1}, S_{in2}, S^{-}_{in2}, S^{+}_{in2} values to the process simulation and MAS. All of these values were generated using compound sinus functions, graphical views of the changes are presented in fig. 8.9 and 8.10. Lastly, LabVIEW simulator simulates process behavior as explained in chapter 3.

During the experiment, MAS capability to reconfigure and adjust used control algorithm according to encountered conditions was tested. Throughout the experiment, set-point values for S_1 and S_2 were periodically changed. For S_1 the set-point value was switched between 2(g/l) and 3(g/l) and for S_2 its corresponding set-point value was switched between 5(g/l) and 6(g/l). Experiment starts with all data available that allows the RAC to take over the control. After 500(h) of simulation, a process control fault - lack of

S^{-}_{in2}, S^{+}_{in2} estimations is introduced which should induce a controller switch by MAS to ensure the best possible control quality.

Experiment results are presented in fig. 8.11 and 8.12. As can be seen in fig. 8.11 and 8.12 from the moment of the simulated fault, character of S_2 noticeably changes and starts to demonstrate a more oscillatory behavior. This is caused by a switch of the controllers that occurred at the fault moment. Up to this time, RAC Control Agent was responsible of setting control values of both D_{11} and D_2. When S^{-}_{in2} and S^{+}_{in2} became unavailable the RAC Control Agent was unable to derive D_2 values anymore. As a result, control over this variable was handed to PI2 Agent which changed the character of D_2 control and S_2 output changes. However, even after the induced fault RAC Control Agent is still responsible of calculating new D_{11} values because the fault does not limit its capability for deriving this CV. Such limited fault event was recognized correctly, MAS performed only necessary switches, and the control over unaffected variables remained in preferred controller's scope of responsibility. This behavior matched author's expectations and proved that MAS built based on prepared instructions is able to control other types of biotechnological processes.

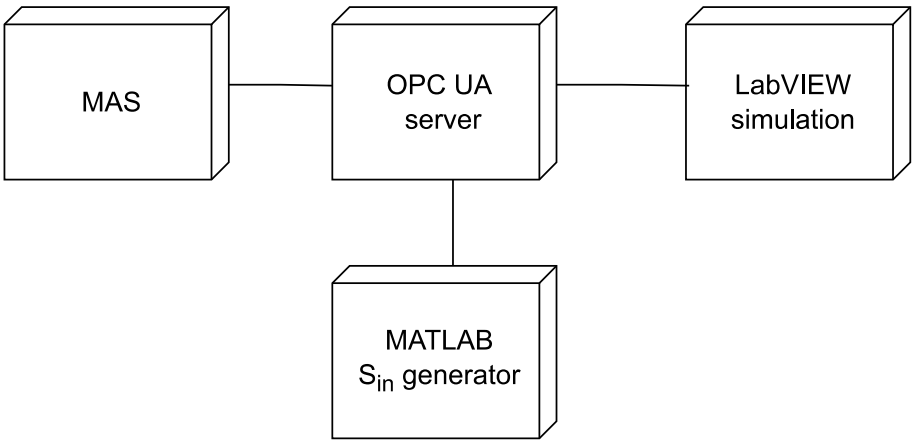

Fig. 8.8 Experiment deployment diagram

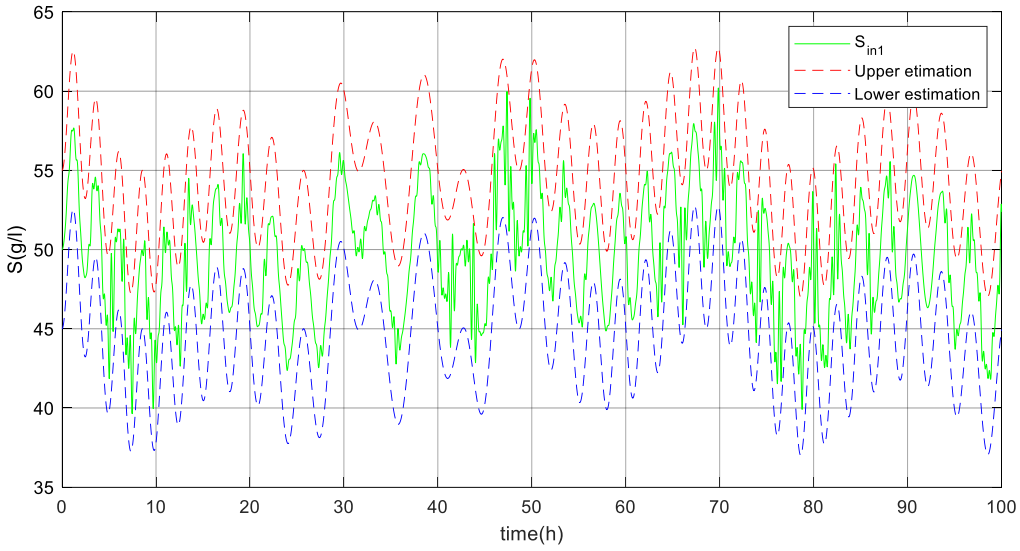

Fig. 8.9 S_{in1} changes over first 100(h) of the experiment

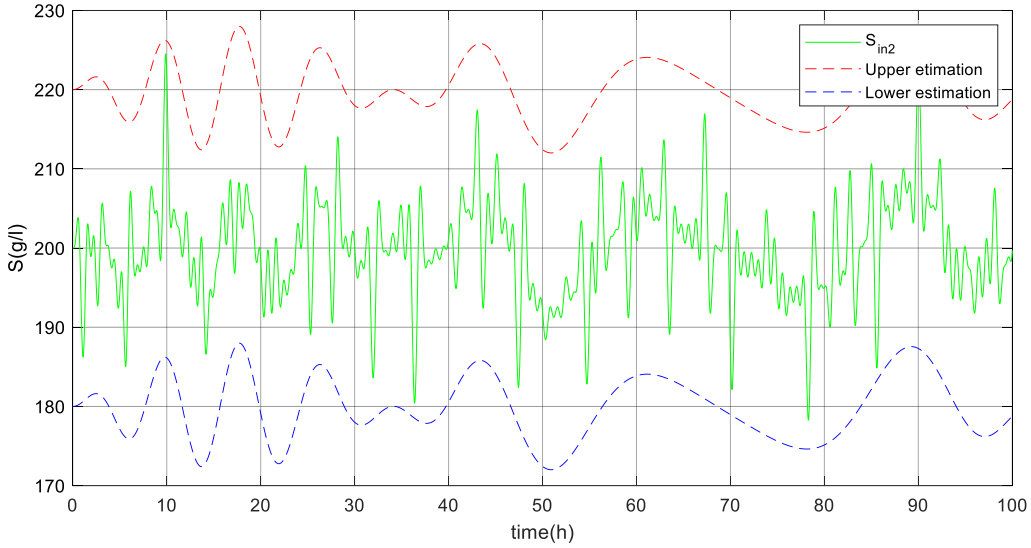

Fig. 8.10 S_{in2} changes over first 100(h) of the experiment

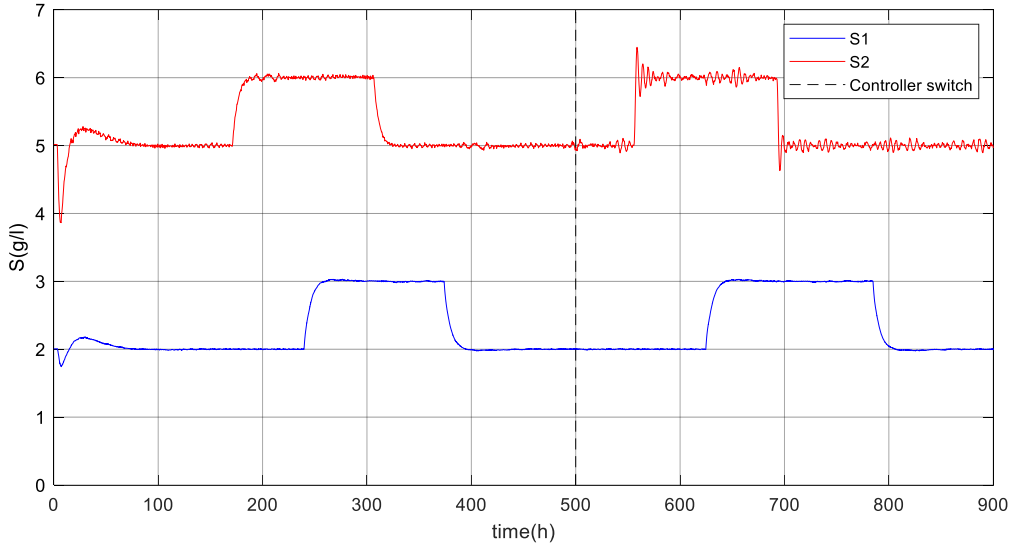

Fig. 8.11 Output values changes during the experiment

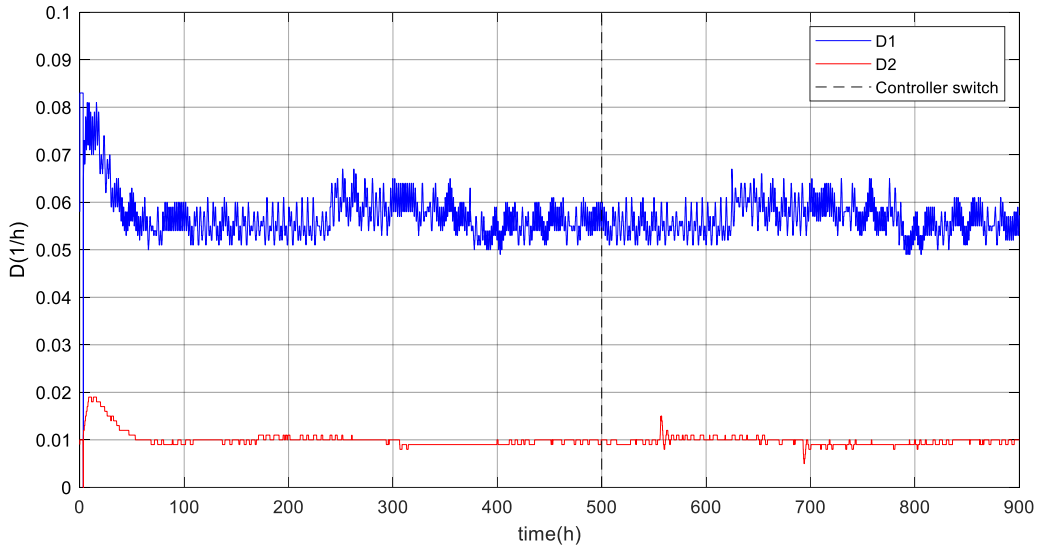

Fig. 8.12 Control values changes during the experiment

8.5. Summary

This chapter presented the verification of prepared MAS and ontology development rules in control of a lactic fermentation process.

Similarly to previous experiments, the prepared MAS also contained multiple control algorithms enclosed inside agents that compete with other agents to take control over the process. However, this time a simpler, priority based control selection approach was chosen, with RAC Control Agent being the one with the highest priority, meaning that its control offers were preferred over other agents.

MAS built based on previously derived instructions was able to the control lactic fermentation process with a satisfying quality. Favored agent was in control whenever possible, but when a fault occurred, MAS was able to adapt to the new circumstances and still provided reliable control values.

Main conclusions from this experiment in the context of this dissertation is the fact that the MAS prepared according to the proposed ontology structure and architecture layout can be successfully applied in control of other types of biotechnological processes Additionally, this experiment was MAS's first test in control of a MIMO systems. It is also important to point out that the prepared ontology schema proved its reusability. During this experiment, the ontology implementation from the previous project was reused, even though a different process was controlled this time. Only minor adjustments to the ontology structure were required. This greatly reduced time and effort needed to develop MAS for this experiment.

9. Final conclusions

This dissertation presented a summary of author's research focused on evaluating usefulness and feasibility of multi-agent systems in biotechnological process control.

The research consisted of four studies divided into two phases. The first phase includes two pilot projects which main objective was an initial evaluation of MAS usefulness for biotechnological process control and general gain of expertise. This expertise was needed to propose multi-agent systems rules for architecture and ontology development aimed at improving MAS availability and lower initial entry threshold.

Second phase was focused on feasibility studies of MAS for biotechnological systems control developed according to proposed development rules.

The research started with the study presented in chapter 4 which was focused on the development of practical MAS implementation with a relatively simple structure. The system was using basic identification and simulation algorithms to realize the control goal limited to precisely reaching only the lower boundary of process output values. Nonetheless it was able to show an idea of how multi-agent systems dedicated for closed loop control can be designed and built. The main conclusions from this study were as follows:

- It proved that non real-time characteristics of agent-oriented tools used in the research are sufficient for reliable control of the aeration process with relatively long sampling time.
- MAS proved its ability to switch control algorithms and adaptability to current conditions. However, in this research adaptability was implemented using a simple priority mechanism.
- MAS design process encountered many difficulties regarding role assignments of specific agents, their relations with each other, communication methods and form of exchanged messages. This issue confirmed statements gathered through the literature review that development of multi agent control systems lacks set of common good practices, rules and hints making this process more difficult and time demanding thus negatively impacting its feasibility. Providing methods to simplify the design of MAS is essential for wider adoption of agent-based control systems.

In response to presented conclusions, following research was focused on preparation of the set of rules for a development of MAS for continuous process control. Results of this research are presented in chapter 5. Proposed set of rules was then verified in MAS which was developed according to the guidelines and then tested during the control of the bioreactor aeration process simulation. Presented validation experiment and its results can be treated as a proof of concept of the proposed architecture design. The result of this research – sets of rules for MAS architecture design and ontology preparation - is a base for following research projects.

Chapter 6 presents continuation of work from the previous chapter focusing on designing an ontology schema for MAS for process control, especially for biotechnological processes. The prepared ontology concept is based on the data and value concepts hierarchy forming an universal core that can be reused in other control systems. Other

classes can be easily modified, added or removed to comply with specific needs of any application. Additionally, the number of classes used in the ontology is still low enough to allow relatively simple manual modifications without a need to use any large-scale tools dedicated for ontologies.

Chapters 7 and 8 present results from the second phase of the research, where MAS systems for biotechnological system control undergone feasibility studies. Firstly, as described in the chapter 7, rules of MAS development alongside the proposed ontology were tested in the development of control system for the real process plant. The task was to prepare and verify a MAS for dissolved oxygen concentration control in an activated sludge wastewater treatment process in a multiple-input control setup. The system should be able to alter MAS configuration during runtime through additions and removals of specific agents. MAS prepared according to the proposed ontology structure and the architecture layout was successfully applied in control of the complex biotechnological process which proved usability of MAS architecture layout. Reused base implementation of the ontology significantly reduced time needed for preparation of system's knowledge base. MAS by itself was able to effectively control the aeration process outperforming a complex, conventional control algorithm, and display self-diagnostic and reconfigurability behaviors. This proved that with enough knowledge and preparation, a MAS based solution might be a competitive alternative to conventionally built systems. Suggested design of MAS for model-based predictive control of continuous processes was successfully applied to control of the real continuous process using multiple actuators and various types of control laws.

Lastly, chapter 8 described verification of prepared MAS and ontology development rules in control of a different biotechnological process i.e. a lactic fermentation process. MAS built based on previously derived instructions was able to control the lactic fermentation process with satisfying quality. Verification proved that MAS prepared according to the proposed ontology structure and the architecture layout can be successfully applied in control of other types of biotechnological processes. Additionally, the experiment was a MAS's first test in control of MIMO systems. It is also important to point out that the prepared ontology schema proved its reusability. During the experiment the ontology implementation from previous project was reused, even though a different process was controlled this time. Only minor adjustments to the ontology structure were required. This greatly reduced time and effort needed to develop MAS for this experiment and improved overall MAS feasibility.

In author's opinion presented results prove MAS usefulness in the context of biotechnological processes control. Prepared systems were able to complete stated objectives, meeting author's needs and expectations. In software development it is often challenging to prove an advantage of one software paradigm over others. In most cases the same functionality can be achieved using various approaches, although with different time effort. In case of agent-based approach, initial effort to familiarize oneself with an environment and prepare a program realizing simple functionalities is higher than in e.g. object-oriented programming. However, with enough expertise, available rules and standards, the effort noticeably drops. This can be seen in chapters 7 and 8 where development according to prepared rules and the possibility to reuse already prepared

components reduced time needed to prepare new implementations. Additionally, as presented in the literature review and in e.g. chapter 7, agent-based systems are dedicated for and are the most useful in solving complex problems like reconfiguration, diagnostics and others that can be encountered in control of the biotechnological processes.

As described in the introduction, MAS feasibility is evaluated based on the two questions:

In author's opinion multi-agent systems are capable of controlling biotechnological systems, proving it experimentally in chapters 4,5,7 and 8. The main doubt regarding this statement was whether agent-based systems built with existent frameworks are able to control real-time processes. Investigation from the chapter 2.1 reveals that this topic was already analyzed in the literature. Although most of modern frameworks are not suitable for direct control of real-time process, hierarchical designs are proposed that allow to use these type of applications to control real-time systems. These designs were successfully used in the dissertation.

Answer to the second question is related to the analysis of MAS's usefulness and depends on the specific case. In general, incorporating MAS in simple control loops, with trivial control algorithms and all the necessary data available online will most likely not be beneficial. An initial effort of building an agent-based system will be greater than using other popular programming paradigms e.g. object-oriented, procedural or graphical approaches. However, when the control problem becomes more complex by including e.g. nonlinearities, uncertainties, unavailability of online measurements, necessity to adapt to the changes of the environment etc. the additional effort associated with handling these problems in the agent-based system will probably be much lower than in its counterparts. This relation can be well presented with MAS from the chapter 7, in presented experiments, agents that realize control algorithms are added and removed online. Achieving the same result with other programming paradigm might be possible, but most likely not trivial, in case of multi-agent systems such behavior is a natural result of agent's autonomy and realizing it required only little additional effort.

In author's opinion the provided evidence is sufficient to prove the thesis that using agent-based system built based on a set of development rules for biotechnological processes enables development of control system with reconfigurable structure which is able to adjust to the controlled processes to achieve satisfactory control quality, improving MAS feasibility and proving its usability.

Future research in that field might involve adoption of MAS system into control of other types of biotechnological processes, developing MAS for more complex control tasks like control of whole wastewater treatment process and further analysis of rules for multi-agent systems development.

References

[1] H. S. Nwana, “Software agents: an overview,” The Knowledge Engineering Review, vol. 11, no. 3, pp. 205–244, Sep. 1996, doi: 10.1017/S026988890000789X.

[2] M. Paolucci and R. Sacile, Agent-Based Manufacturing and Control Systems: New Agile Manufacturing Solutions for Achieving Peak Performance. Boca Raton: CRC Press, 2004. doi: 10.1201/9780203492666.

[3] M. d’Inverno and M. Luck, Understanding Agent Systems, second edition. in Springer Series on Agent Technology. Springer-Verlag Berlin Heidelberg, 2004.

[4] T. Selker, “COACH: a teaching agent that learns,” Commun. ACM, vol. 37, no. 7, pp. 92–99, Jul. 1994, doi: 10.1145/176789.176799.

[5] D. C. Smith, A. Cypher, and J. Spohrer, “KidSim: programming agents without a programming language,” Commun. ACM, vol. 37, no. 7, pp. 54–67, Jul. 1994, doi: 10.1145/176789.176795.

[6] D. Riecken, “M: an architecture of integrated agents,” Commun. ACM, vol. 37, no. 7, p. 106, Jul. 1994, doi: 10.1145/176789.176801.

[7] J. M. Bradshaw, “An introduction to software agents,” in Software agents, Cambridge, MA, USA: MIT Press, 1997, pp. 3–46.

[8] P. Maes, “Modeling Adaptive Autonomous Agents,” Artif Life, vol. 1, no. 1_2, pp. 135–162, Oct. 1993, doi: 10.1162/artl.1993.1.1_2.135.

[9] D. Wong, N. Paciorek, and D. Moore, “Java-based mobile agents,” Commun. ACM, vol. 42, no. 3, p. 92, Mar. 1999, doi: 10.1145/295685.295717.

[10] I. F. Imam and Y. Kodratoff, “Intelligent Adaptive Agents: A Highlight of the Field and the AAAI-96 Workshop,” AI Magazine, vol. 18, no. 3, pp. 75–75, Sep. 1997, doi: 10.1609/aimag.v18i3.1307.

[11] M. Wooldridge and N. R. Jennings, “Intelligent agents: theory and practice,” The Knowledge Engineering Review, vol. 10, no. 2, pp. 115–152, Jun. 1995, doi: 10.1017/S0269888900008122.

[12] M. Woolridge and M. J. Wooldridge, Introduction to Multiagent Systems. USA: John Wiley & Sons, Inc., 2001.

[13] W. Sabra, D. Dietz, D. Tjahjasari, and A.-P. Zeng, “Biosystems analysis and engineering of microbial consortia for industrial biotechnology,” Engineering in Life Sciences, vol. 10, no. 5, pp. 407–421, 2010, doi: 10.1002/elsc.201000111.

[14] G. Festel, C. Detzel, and R. Maas, “Industrial biotechnology -- Markets and industry structure.,” Journal of Commercial Biotechnology, vol. 18, no. 1, pp. 11–21, Jan. 2012, doi: 10.5912/jcb478.

[15] A. J. J. Straathof, S. A. Wahl, K. R. Benjamin, R. Takors, N. Wierckx, and H. J. Noorman, “Grand Research Challenges for Sustainable Industrial Biotechnology,” Trends in Biotechnology, vol. 37, no. 10, pp. 1042–1050, Oct. 2019, doi: 10.1016/j.tibtech.2019.04.002.

[16] W. L. Tang and H. Zhao, “Industrial biotechnology: Tools and applications,” Biotechnology Journal, vol. 4, no. 12, pp. 1725–1739, 2009, doi: 10.1002/biot.200900127.

[17] J. M. Otero and J. Nielsen, “Industrial systems biology,” Biotechnology and Bioengineering, vol. 105, no. 3, pp. 439–460, 2010, doi: 10.1002/bit.22592.

[18] M. Metzger and G. Polakow, “A Survey on Applications of Agent Technology in Industrial Process Control,” IEEE Transactions on Industrial Informatics, vol. 7, no. 4, pp. 570–581, Nov. 2011, doi: 10.1109/TII.2011.2166781.

[19] M. Kolankowski, M. Banach, R. Piotrowski, and T. Ujazdowski, “A New Approach to Designing Control of Dissolved Oxygen and Aeration System in Sequencing Batch Reactor by Applied Backstepping Control Algorithm,” Acta Mechanica et Automatica, vol. 17, no. 4, pp. 605–612, Dec. 2023, doi: 10.2478/ama-2023-0070.

[20] R. Piotrowski, M. Wonia, and A. Wonia, “Stochastic optimisation algorithm for optimisation of controller parameters for control of dissolved oxygen in wastewater treatment plant,” Journal of Water Process Engineering, vol. 51, p. 103357, Feb. 2023, doi: 10.1016/j.jwpe.2022.103357.

[21] F. Hernandez-del-Olmo, E. Gaudioso, and A. Nevado, “Autonomous Adaptive and Active Tuning Up of the Dissolved Oxygen Setpoint in a Wastewater Treatment Plant Using Reinforcement Learning,” IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 42, no. 5, pp. 768–774, Sep. 2012, doi: 10.1109/TSMCC.2011.2162401.

[22] Y.-C. Bo and X. Zhang, “Online adaptive dynamic programming based on echo state networks for dissolved oxygen control,” Applied Soft Computing, vol. 62, pp. 830–839, Jan. 2018, doi: 10.1016/j.asoc.2017.09.015.

[23] P. Laszczyk, “Predictive functional control of dissolved oxygen with online estimation of oxygene uptake rate,” in 2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR), Aug. 2015, pp. 602–607. doi: 10.1109/MMAR.2015.7283943.

[24] H. Han, L. Zhang, and J. Qiao, “Data-Based Predictive Control for Wastewater Treatment Process,” IEEE Access, vol. 6, pp. 1498–1512, 2018, doi: 10.1109/ACCESS.2017.2779175.

[25] K. Duzinkiewicz, M. A. Brdys, W. Kurek, and R. Piotrowski, “Genetic Hybrid Predictive Controller for Optimized Dissolved-Oxygen Tracking at Lower Control Level,” IEEE Transactions on Control Systems Technology, vol. 17, no. 5, pp. 1183–1192, Sep. 2009, doi: 10.1109/TCST.2008.2004499.

[26] K. Stebel, J. Pospiech, W. Nocon, J. Czeczot, and P. Skupin, “Boundary-Based Predictive Controller and Its Application to Control of Dissolved Oxygen Concentration in Activated Sludge Bioreactor,” IEEE Transactions on Industrial Electronics, vol. 69, no. 10, pp. 10541–10551, Oct. 2022, doi: 10.1109/TIE.2021.3123629.

[27] C. Chen, H. Han, H. Sun, H. Yang, and J. Qiao, “Multi-Objective Integrated Robust Optimal Control for Wastewater Treatment Processes,” IEEE Transactions on Automation Science and Engineering, vol. 21, no. 2, pp. 1380–1391, Apr. 2024, doi: 10.1109/TASE.2023.3240497.

[28] S. Fu, H. Sun, Z. Liu, H. Han, and Y. Zhang, “Multimodel Predictive Control for Wastewater Treatment Process with Multirate Sampling Intervals,” in 2023 IEEE 13th International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER), Jul. 2023, pp. 610–615. doi: 10.1109/CYBER59472.2023.10256568.

[29] H. Han, C. Feng, H. Sun, and J. Qiao, “Self-Organizing Fuzzy Terminal Sliding Mode Control for Wastewater Treatment Processes,” IEEE Transactions on Automation Science and Engineering, vol. 21, no. 4, pp. 5421–5433, Oct. 2024, doi: 10.1109/TASE.2023.3311768.

[30] R. Piotrowski, M. A. Brdys, K. Konarczak, K. Duzinkiewicz, and W. Chotkowski, “Hierarchical dissolved oxygen control for activated sludge processes,” Control Engineering Practice, vol. 16, no. 1, pp. 114–131, Jan. 2008, doi: 10.1016/j.conengprac.2007.04.005.

[31] R. Piotrowski, H. Sawicki, and K. Żuk, “Novel hierarchical nonlinear control algorithm to improve dissolved oxygen control in biological WWTP,” Journal of Process Control, vol. 105, pp. 78–87, Sep. 2021, doi: 10.1016/j.jprocont.2021.07.009.

[32] T. Chistiakova, T. Wigren, and B. Carlsson, “Combined \mathcalL_2 -Stable Feedback and Feedforward Aeration Control in a Wastewater Treatment Plant,” IEEE Transactions on Control Systems Technology, vol. 28, no. 3, pp. 1017–1024, May 2020, doi: 10.1109/TCST.2019.2891410.

[33] F. J. dos Santos Silva, S. Y. C. Catunda, C. E. T. Dorea, A. C. van Haandel, and H. R. dos Santos, “Oxygen Uptake Rate Measurement Using Kalman Filter and PWM Control in Activated Sludge Systems,” IEEE Transactions on Instrumentation and Measurement, vol. 68, no. 11, pp. 4493–4501, Nov. 2019, doi: 10.1109/TIM.2018.2886941.

[34] A. O. Ojo and O. de Smidt, “Lactic Acid: A Comprehensive Review of Production to Purification,” Processes, vol. 11, no. 3, p. 688, Mar. 2023, doi: 10.3390/pr11030688.

[35] E. Abedi and S. M. B. Hashemi, “Lactic acid production – producing microorganisms and substrates sources-state of art,” Heliyon, vol. 6, no. 10, Oct. 2020, doi: 10.1016/j.heliyon.2020.e04974.

[36] C. Ben Youssef, V. Guillou, and A. Olmos-Dichara, “Modelling and adaptive control strategy in a lactic fermentation process,” Control Engineering Practice, vol. 8, no. 11, pp. 1297–1307, Nov. 2000, doi: 10.1016/S0967-0661(00)00061-7.

[37] E. Petre, D. Selișteanu, and M. Roman, “Nonlinear robust adaptive control strategies for a lactic fermentation process,” Journal of Chemical Technology & Biotechnology, vol. 93, no. 2, pp. 518–526, 2018, doi: 10.1002/jctb.5383.

[38] K. Gonzalez et al., “Regulation of lactic acid concentration in its bioproduction from wheat flour,” Control Engineering Practice, vol. 54, pp. 202–213, Sep. 2016, doi: 10.1016/j.conengprac.2016.06.006.

[39] K. Gonzalez et al., “Feedback linearizing controller coupled to an Unscented Kalman filter for lactic acid regulation,” in 2015 19th International Conference on System Theory, Control and Computing (ICSTCC), Oct. 2015, pp. 219–224. doi: 10.1109/ICSTCC.2015.7321296.

[40] K. V. Gonzalez et al., “Adaptive Control of Lactic Acid Production Process from Wheat Flour,” IFAC-PapersOnLine, vol. 48, no. 8, pp. 1087–1092, Jan. 2015, doi: 10.1016/j.ifacol.2015.09.113.

[41] A. M. Gujarathi, S. P. Patel, and B. A. Siyabi, “Insight into evolutionary optimization approach of batch and fed-batch fermenters for lactic acid production,” Digital Chemical Engineering, vol. 8, p. 100105, Sep. 2023, doi: 10.1016/j.dche.2023.100105.

[42] P. A. López-Pérez, M. López-López, C. A. Núñez-Colín, H. Mukhtar, R. Aguilar-López, and V. Peña-Caballero, “A novel nonlinear sliding mode observer to estimate biomass for lactic acid production,” Chemical Product and Process Modeling, vol. 18, no. 4, pp. 565–580, Aug. 2023, doi: 10.1515/cppm-2021-0074.

[43] Y. Wang et al., “A Distributed Control Scheme of Microgrids in Energy Internet Paradigm and Its Multisite Implementation,” IEEE Transactions on Industrial Informatics, vol. 17, no. 2, pp. 1141–1153, Feb. 2021, doi: 10.1109/TII.2020.2976830.

[44] Z. Zhang, D. Yue, and C.-X. Dou, “DMPC-Based Coordinated Voltage Control for Integrated Hybrid Energy System,” IEEE Transactions on Industrial Informatics, vol. 17, no. 10, pp. 6786–6797, Oct. 2021, doi: 10.1109/TII.2020.3046633.

[45] 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, pp. 1–6. doi: 10.1109/ISAP.2015.7325531.

[46] M. Manbachi and M. Ordonez, “Intelligent Agent-Based Energy Management System for Islanded AC–DC Microgrids,” IEEE Transactions on Industrial Informatics, vol. 16, no. 7, pp. 4603–4614, Jul. 2020, doi: 10.1109/TII.2019.2945371.

[47] M. B. Menhaj, A. Fakharian, P. Qaderi-Baban, and M. Dosaranian-Moghadam, “Intelligent multi-agent system for DC microgrid energy coordination control,” Bulletin of the Polish Academy of Sciences: Technical Sciences; 2019; 67; No. 4; 741-748, 2019, Accessed: Nov. 26, 2023. [Online]. Available: https://journals.pan.pl/dlibra/publication/130183/edition/113667

[48] C. Dou, D. Yue, X. Li, and Y. Xue, “MAS-Based Management and Control Strategies for Integrated Hybrid Energy System,” IEEE Transactions on Industrial Informatics, vol. 12, no. 4, pp. 1332–1349, Aug. 2016, doi: 10.1109/TII.2016.2569506.

[49] X. Gao, K. W. Chan, S. Xia, X. Zhang, K. Zhang, and J. Zhou, “A Multiagent Competitive Bidding Strategy in a Pool-Based Electricity Market With Price-Maker Participants of WPPs and EV Aggregators,” IEEE Transactions on Industrial Informatics, vol. 17, no. 11, pp. 7256–7268, Nov. 2021, doi: 10.1109/TII.2021.3055817.

[50] M. M. Esfahani, A. Hariri, and O. A. Mohammed, “A Multiagent-Based Game-Theoretic and Optimization Approach for Market Operation of Multimicrogrid Systems,” IEEE Transactions on Industrial Informatics, vol. 15, no. 1, pp. 280–292, Jan. 2019, doi: 10.1109/TII.2018.2808183.

[51] A. Hussain, V.-H. Bui, and H.-M. Kim, “An Effort-Based Reward Approach for Allocating Load Shedding Amount in Networked Microgrids Using Multiagent System,” IEEE Transactions on Industrial Informatics, vol. 16, no. 4, pp. 2268–2279, Apr. 2020, doi: 10.1109/TII.2019.2929284.

[52] E. Shirazi and S. Jadid, “Autonomous Self-Healing in Smart Distribution Grids Using agent Systems,” IEEE Transactions on Industrial Informatics, vol. 15, no. 12, pp. 6291–6301, Dec. 2019, doi: 10.1109/TII.2018.2889741.

[53] S. A., R. Kumar, and R. C. Bansal, “Multiagent-Based Autonomous Energy Management System With Self-Healing Capabilities for a Microgrid,” IEEE Transactions on Industrial Informatics, vol. 15, no. 12, pp. 6280–6290, Dec. 2019, doi: 10.1109/TII.2018.2889692.

[54] N. R. Jennings and S. Bussmann, “Agent-based control systems: Why are they suited to engineering complex systems?,” IEEE Control Systems Magazine, vol. 23, no. 3, pp. 61–73, Jun. 2003, doi: 10.1109/MCS.2003.1200249.

[55] V. Mařík and J. Lažanský, “Industrial applications of agent technologies,” Control Engineering Practice, vol. 15, no. 11, pp. 1364–1380, Nov. 2007, doi: 10.1016/j.conengprac.2006.10.001.

[56] M. Bennulf, F. Danielsson, B. Svensson, and B. Lennartson, “Goal-Oriented Process Plans in a Multiagent System for Plug & Produce,” IEEE Transactions on Industrial Informatics, vol. 17, no. 4, pp. 2411–2421, Apr. 2021, doi: 10.1109/TII.2020.2994032.

[57] T. Arai, Y. Aiyama, Y. Maeda, M. Sugi, and J. Ota, “Agile Assembly System by ‘Plug and Produce,’” CIRP Annals, vol. 49, no. 1, pp. 1–4, Jan. 2000, doi: 10.1016/S0007-8506(07)62883-2.

[58] J. W. Park, M. Shin, and D. Y. Kim, “An Extended Agent Communication Framework for Rapid Reconfiguration of Distributed Manufacturing Systems,” IEEE Transactions on Industrial Informatics, vol. 15, no. 7, pp. 3845–3855, Jul. 2019, doi: 10.1109/TII.2018.2883409.

[59] W. Lepuschitz, A. Zoitl, M. Vallée, and M. Merdan, “Toward Self-Reconfiguration of Manufacturing Systems Using Automation Agents,” IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), vol. 41, no. 1, pp. 52–69, Jan. 2011, doi: 10.1109/TSMCC.2010.2059012.

[60] 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), Oct. 2001, pp. 481–488 vol.2. doi: 10.1109/ETFA.2001.997722.

[61] P. Leitão, N. Rodrigues, C. Turrin, and A. Pagani, “Multiagent System Integrating Process and Quality Control in a Factory Producing Laundry Washing Machines,” IEEE Transactions on Industrial Informatics, vol. 11, no. 4, pp. 879–886, Aug. 2015, doi: 10.1109/TII.2015.2431232.

[62] R. Sikorski, “Flexible multi-agent system for mobile robot group control,” ELECTROTECHNICAL REVIEW, vol. 1, no. 12, pp. 254–258, Dec. 2019, doi: 10.15199/48.2019.12.57.

[63] F. Logi and S. G. Ritchie, “A multi-agent architecture for cooperative inter-jurisdictional traffic congestion management,” Transportation Research Part C: Emerging Technologies, vol. 10, no. 5, pp. 507–527, Oct. 2002, doi: 10.1016/S0968-090X(02)00033-5.

[64] M. A. S. Kamal, C. P. Tan, T. Hayakawa, S.-I. Azuma, and J.-I. Imura, “Control of Vehicular Traffic at an Intersection Using a Cyber-Physical Multiagent Framework,” IEEE Transactions on Industrial Informatics, vol. 17, no. 9, pp. 6230–6240, Sep. 2021, doi: 10.1109/TII.2021.3051961.

[65] P.-L. Gregoire, C. Desjardins, J. Laumonier, and B. Chaib-draa, “Urban Traffic Control Based on Learning Agents,” in 2007 IEEE Intelligent Transportation Systems Conference, Sep. 2007, pp. 916–921. doi: 10.1109/ITSC.2007.4357719.

[66] D. A. Roozemond, “USING AUTONOMOUS INTELLIGENT AGENTS FOR URBAN TRAFFIC CONTROL SYSTEMS,” presented at the PROCEEDINGS OF 6TH WORLD CONGRESS ON INTELLIGENT TRANSPORT SYSTEMS (ITS), HELD TORONTO, CANADA, NOVEMBER 8-12, 1999, 1999. Accessed: Aug. 14, 2025. [Online]. Available: https://trid.trb.org/View/695298

[67] A. Cuppari, P. L. Guida, M. Martelli, V. Mascardi, and F. Zini, “Prototyping freight trains traffic management using multi-agent systems,” in Proceedings 1999 International Conference on Information Intelligence and Systems (Cat. No.PR00446), Oct. 1999, pp. 646–653. doi: 10.1109/ICIIS.1999.810360.

[68] J. Blum and A. Eskandarian, “Enhancing intelligent agent collaboration for flow optimization of railroad traffic,” Transportation Research Part A: Policy and Practice, vol. 36, no. 10, pp. 919–930, Dec. 2002, doi: 10.1016/S0965-8564(02)00002-2.

[69] B. N. Iordanova, “Air traffic knowledge management policy,” European Journal of Operational Research, vol. 146, no. 1, pp. 83–100, Apr. 2003, doi: 10.1016/S0377-2217(02)00151-0.

[70] S. Wollkind, J. Valasek, and T. Ioerger, “Automated Conflict Resolution for Air Traffic Management Using Cooperative Multiagent Negotiation,” in AIAA Guidance, Navigation, and Control Conference and Exhibit, American Institute of Aeronautics and Astronautics. doi: 10.2514/6.2004-4992.

[71] M. Pěchouček, D. Šišlák, D. Pavlíček, and M. Uller, “Autonomous agents for air-traffic deconfliction,” in Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, in AAMAS ’06. New York, NY, USA: Association for Computing Machinery, May 2006, pp. 1498–1505. doi: 10.1145/1160633.1160925.

[72] D. Weyns, T. Holvoet, and A. Helleboogh, “Anticipatory Vehicle Routing using Delegate Multi-Agent Systems,” in 2007 IEEE Intelligent Transportation Systems Conference, Sep. 2007, pp. 87–93. doi: 10.1109/ITSC.2007.4357809.

[73] J. Lind and K. Fischer, “Transportation Scheduling and Simulation in a Railroad Scenario: A Multi-Agent Approach,” in Logistik Management, H. Kopfer and C. Bierwirth, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 171–183. doi: 10.1007/978-3-642-60184-2_15.

[74] K. M. Hanga and Y. Kovalchuk, “Machine learning and multi-agent systems in oil and gas industry applications: A survey,” Computer Science Review, vol. 34, p. 100191, Nov. 2019, doi: 10.1016/j.cosrev.2019.08.002.

[75] M. A. Ramírez, J. Dávila, and E. C. Morles, “Intelligent supervision based on Multi-Agent Systems: 13th WSEAS International Conference on Systems - Held as part of the 13th WSEAS CSCC Multiconference,” Proceedings of the 13th WSEAS International Conference on Systems - Held as part of the 13th WSEAS CSCC Multiconference, pp. 498–505, 2009.

[76] V. L. C. de Oliveira, A. P. M. Tanajura, and H. A. Lepikson, “A Multi-agent System for Oil Field Management,” IFAC Proceedings Volumes, vol. 46, no. 7, pp. 35–40, May 2013, doi: 10.3182/20130522-3-BR-4036.00010.

[77] L. L. Mikkelsen and B. N. Jørgensen, “Towards Intelligent Optimization of Offshore Oil and Gas Production Using Multi-agent Software Systems,” presented at the SPE Western Regional Meeting, OnePetro, Mar. 2012. doi: 10.2118/153815-MS.

[78] M. K. Ahmed, A. Y. Umar, and M. S. Bute, “Multi-agent based architectural framework for the prevention and control of oil pipeline vandalism,” in 2017 International Conference on Computing Networking and Informatics (ICCNI), Oct. 2017, pp. 1–8. doi: 10.1109/ICCNI.2017.8123808.

[79] 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, vol. 9, no. 3, pp. 233–248, Mar. 2001, doi: 10.1016/S0967-0661(00)00111-8.

[80] P. Davidsson and M. Boman, “Distributed monitoring and control of office buildings by embedded agents,” Information Sciences, vol. 171, no. 4, pp. 293–307, May 2005, doi: 10.1016/j.ins.2004.09.007.

[81] B. Asare-Bediako, W. L. Kling, and P. F. Ribeiro, “Multi-agent system architecture for smart home energy management and optimization,” in IEEE PES ISGT Europe 2013, Oct. 2013, pp. 1–5. doi: 10.1109/ISGTEurope.2013.6695331.

[82] J. Duan and F. Lin, “Research of intelligent building control using an agent-based approach,” in 2008 6th IEEE International Conference on Industrial Informatics, Jul. 2008, pp. 991–994. doi: 10.1109/INDIN.2008.4618246.

[83] P. Davidsson and M. Boman, “Saving Energy and Providing Value Added Services in Intelligent Buildings: A MAS Approach,” in Agent Systems, Mobile Agents, and Applications, D. Kotz and F. Mattern, Eds., Berlin, Heidelberg: Springer, 2000, pp. 166–177. doi: 10.1007/978-3-540-45347-5_14.

[84] F.-L. Lian, J. K. Yook, D. M. Tilbury, and J. Moyne, “Network architecture and communication modules for guaranteeing acceptable control and communication performance for networked multi-agent systems,” IEEE Transactions on Industrial Informatics, vol. 2, no. 1, pp. 12–24, Feb. 2006, doi: 10.1109/TII.2005.857611.

[85] J. Xu, Z. Dziong, Y. Luxin, Z. Huang, P. Xu, and A. Cabani, “Intelligent multi-agent based C-RAN architecture for 5G radio resource management,” Computer Networks, vol. 180, p. 107418, Oct. 2020, doi: 10.1016/j.comnet.2020.107418.

[86] Z. Zuo, Q.-L. Han, B. Ning, X. Ge, and X.-M. Zhang, “An Overview of Recent Advances in Fixed-Time Cooperative Control of Multiagent Systems,” IEEE Transactions on Industrial Informatics, vol. 14, no. 6, pp. 2322–2334, Jun. 2018, doi: 10.1109/TII.2018.2817248.

[87] K.-K. Oh, M.-C. Park, and H.-S. Ahn, “A survey of multi-agent formation control,” Automatica, vol. 53, pp. 424–440, Mar. 2015, doi: 10.1016/j.automatica.2014.10.022.

[88] T. Han, Z.-H. Guan, M. Chi, B. Hu, T. Li, and X.-H. Zhang, “Multi-formation control of nonlinear leader-following multi-agent systems,” ISA Transactions, vol. 69, pp. 140–147, Jul. 2017, doi: 10.1016/j.isatra.2017.05.003.

[89] B. Wang, W. Chen, B. Zhang, Y. Zhao, and P. Shi, “Cooperative Control-Based Task Assignments for Multiagent Systems With Intermittent Communication,” IEEE Transactions on Industrial Informatics, vol. 17, no. 10, pp. 6697–6708, Oct. 2021, doi: 10.1109/TII.2020.3044950.

[90] Y. Gao, Z. Shang, and A. Kokossis, “Agent-based intelligent system development for decision support in chemical process industry,” Expert Systems with Applications, vol. 36, no. 8, pp. 11099–11107, Oct. 2009, doi: 10.1016/j.eswa.2009.02.078.

[91] S. Natarajan and R. Srinivasan, “Implementation of multi agents based system for process supervision in large-scale chemical plants,” Computers & Chemical Engineering, vol. 60, pp. 182–196, Jan. 2014, doi: 10.1016/j.compchemeng.2013.08.012.

[92] H. Wang and Y. Zhang, “Multi-Agent Based Chemical Plant Process Monitoring and Management System,” in 2008 4th International Conference on Wireless Communications, Networking and Mobile Computing, Oct. 2008, pp. 1–4. doi: 10.1109/WiCom.2008.2846.

[93] Y. Seng Ng and R. Srinivasan, “Multi-agent based collaborative fault detection and identification in chemical processes,” Engineering Applications of Artificial Intelligence, vol. 23, no. 6, pp. 934–949, Sep. 2010, doi: 10.1016/j.engappai.2010.01.026.

[94] 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, August 16–19, 2006, D.-S. Huang, K. Li, and G. W. Irwin, Eds., Berlin, Heidelberg: Springer, 2006, pp. 319–330. doi: 10.1007/978-3-540-37256-1_39.

[95] Z. Hongwei, Q. Yiming, Z. Jiye, and Z. Yuanheng, “Based on Multi-agent Model for Grinding Process Control Research,” in 2010 Fifth International Conference on Frontier of Computer Science and Technology, Aug. 2010, pp. 576–581. doi: 10.1109/FCST.2010.41.

[96] H.-K. Kim and S. Kim, “Multi-agent Based Incineration Process Control System with Qualitative Model,” in Agent Computing and Multi-Agent Systems, A. Ghose, G. Governatori, and R. Sadananda, Eds., Berlin, Heidelberg: Springer, 2009, pp. 372–379. doi: 10.1007/978-3-642-01639-4_33.

[97] M. Francisco, Y. Mezquita, S. Revollar, P. Vega, and J. F. De Paz, “Multi-agent distributed model predictive control with fuzzy negotiation,” Expert Systems with Applications, vol. 129, pp. 68–83, Sep. 2019, doi: 10.1016/j.eswa.2019.03.056.

[98] K. Fischer, S. Jacobi, C. Diehl, and C. Theis, “Multiagent technologies for steel production and control,” in Proceedings. IEEE/WIC/ACM International Conference on Intelligent Agent Technology, 2004. (IAT 2004)., Sep. 2004, pp. 555–558. doi: 10.1109/IAT.2004.1343017.

[99] J. Boes and F. Migeon, “Self-organizing multi-agent systems for the control of complex systems,” Journal of Systems and Software, vol. 134, pp. 12–28, Dec. 2017, doi: 10.1016/j.jss.2017.08.038.

[100] 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), Jun. 2015, pp. 375–380. doi: 10.1109/PC.2015.7169992.

[101] M. Sànchez, U. Cortés, J. Lafuente, I. R. Roda, and M. Poch, “DAI-DEPUR: an integrated and distributed architecture for wastewater treatment plants supervision,” Artificial Intelligence in Engineering, vol. 10, no. 3, pp. 275–285, Aug. 1996, doi: 10.1016/0954-1810(96)00004-0.

[102] J. Baeza, D. Gabriel, J. Béjar, and J. Lafuente, “A Distributed Control System Based on Agent Architecture for Wastewater Treatment,” Computer-Aided Civil and Infrastructure Engineering, vol. 17, no. 2, pp. 93–103, 2002, doi: 10.1111/1467-8667.00258.

[103] S. Videau, C. Bernon, P. Glize, and J.-L. Uribelarrea, “Controlling Bioprocesses Using Cooperative Self-organizing Agents,” in Advances on Practical Applications of Agents and Multiagent Systems, Y. Demazeau, M. Pěchouček, J.

M. Corchado, and J. B. Pérez, Eds., in Advances in Intelligent and Soft Computing. Berlin, Heidelberg: Springer, 2011, pp. 141–150. doi: 10.1007/978-3-642-19875-5_19.

[104] G. Polaków and M. Metzger, "Agent-Based Control System for Sustainable Wastewater Treatment Process," in Multi-disciplinary Trends in Artificial Intelligence, C. Sombattheera, N. K. Loi, R. Wankar, and T. Quan, Eds., Berlin, Heidelberg: Springer, 2012, pp. 202–213. doi: 10.1007/978-3-642-35455-7_19.

[105] D. Choinski, M. Metzger, and W. Nocon, "Voting in Multi-Agent System for Improvement of Partial Observations," in Agent and Multi-Agent Systems: Technologies and Applications, J. O'Shea, N. T. Nguyen, K. Crockett, R. J. Howlett, and L. C. Jain, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2011, pp. 353–362. doi: 10.1007/978-3-642-22000-5_37.

[106] M. Falco and G. Robiolo, "A Systematic Literature Review in Multi-Agent Systems: Patterns and Trends," in 2019 XLV Latin American Computing Conference (CLEI), Sep. 2019, pp. 1–10. doi: 10.1109/CLEI47609.2019.235098.

[107] 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, vol. 14, no. 3, pp. 18–32, Sep. 2020, doi: 10.1109/MIE.2019.2962225.

[108] S. Karnouskos, L. Ribeiro, P. Leitão, A. Lüder, and B. Vogel-Heuser, "Key Directions for Industrial Agent Based Cyber-Physical Production Systems," in 2019 IEEE International Conference on Industrial Cyber Physical Systems (ICPS), May 2019, pp. 17–22. doi: 10.1109/ICPHYS.2019.8780360.

[109] 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, N. T. Nguyen, A. Grzech, R. J. Howlett, and L. C. Jain, Eds., Berlin, Heidelberg: Springer, 2007, pp. 655–664. doi: 10.1007/978-3-540-72830-6_68.

[110] "Jade Site | Java Agent DEvelopment Framework." Accessed: Aug. 15, 2025. [Online]. Available: https://jade.tilab.com/

[111] F. Bergenti, G. Caire, S. Monica, and A. Poggi, "The first twenty years of agent-based software development with JADE," Auton Agent Multi-Agent Syst, vol. 34, no. 2, p. 36, May 2020, doi: 10.1007/s10458-020-09460-z.

[112] K. Kravari and N. Bassiliades, "A Survey of Agent Platforms," Journal of Artificial Societies and Social Simulation, vol. 18, no. 1, pp. 1–11, 2015, doi: 10.18564/jasss.2661.

[113] J. P. Müller and K. Fischer, "Application Impact of Multi-agent Systems and Technologies: A Survey," in Agent-Oriented Software Engineering: Reflections on Architectures, Methodologies, Languages, and Frameworks, O. Shehory and A. Sturm, Eds., Berlin, Heidelberg: Springer, 2014, pp. 27–53. doi: 10.1007/978-3-642-54432-3_3.

[114] P. D. O'Brien and R. C. Nicol, "FIPA — Towards a Standard for Software Agents," BT Technology Journal, vol. 16, no. 3, pp. 51–59, Jul. 1998, doi: 10.1023/A:1009621729979.

[115] "Wade Site | Workflows and Agent DEvelopment Environment." Accessed: Aug. 15, 2025. [Online]. Available: https://jade.tilab.com/wadeproject/

[116] I. Nunes, ingridnunes/bdi4jade. (Apr. 21, 2025). Java. Accessed: Aug. 15, 2025. [Online]. Available: https://github.com/ingridnunes/bdi4jade

[117] "Active Components." Accessed: Aug. 15, 2025. [Online]. Available: https://www.activecomponents.org/#/project/news

[118] aiagents/jadescript. (Jan. 21, 2024). Java. aiagents. Accessed: Aug. 18, 2025. [Online]. Available: https://github.com/aiagents/jadescript

[119] "ASTRA Programming Language." Accessed: Aug. 19, 2025. [Online]. Available: https://guide.astralanguage.com/en/latest/

[120] "JACK," AOS Group. Accessed: Aug. 15, 2025. [Online]. Available: https://aosgrp.com.au/jack/

[121] S. Team, "SPADE - Python Multi-Agent Framework | Smart Agent Development Environment." Accessed: Aug. 15, 2025. [Online]. Available: https://github.com/javipalanca/spade

[122] "masr / bspl · GitLab," GitLab. Accessed: Aug. 15, 2025. [Online]. Available: https://gitlab.com/masr/bspl

[123] "Pierre DUBAILLAY / piaf · GitLab," GitLab. Accessed: Aug. 15, 2025. [Online]. Available: https://gitlab.com/ornythorinque/piaf

[124] Z. Wrona et al., "Overview of Software Agent Platforms Available in 2023," Information, vol. 14, no. 6, p. 348, Jun. 2023, doi: 10.3390/info14060348.

[125] 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), Aug. 2016, pp. 571–576. doi: 10.1109/MMAR.2016.7575199.

[126] 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), Jul. 2018, pp. 121–126. doi: 10.1109/INDIN.2018.8471927.

[127] W. Lepuschitz, G. Koppensteiner, M. Barta, T. T. Nguyen, and C. Reinprecht, "Implementation of automation agents for batch process automation," in 2010 IEEE International Conference on Industrial Technology, Mar. 2010, pp. 524–529. doi: 10.1109/ICIT.2010.5472745.

[128] H. Kaindl, M. Vallee, and E. Arnautovic, "Self-Representation for Self-Configuration and Monitoring in Agent-Based Flexible Automation Systems," IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 43, no. 1, pp. 164–175, Jan. 2013, doi: 10.1109/TSMCA.2012.2192922.

[129] P. Leitão, S. Karnouskos, L. Ribeiro, P. Moutis, J. Barbosa, and Thomas. I. Strasser, "Integration Patterns for Interfacing Software Agents with Industrial Automation Systems," in IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society, Oct. 2018, pp. 2908–2913. doi: 10.1109/IECON.2018.8591641.

[130] G. Tchobanoglous, F. L. Burton and H. D. Stensel, "WastewaterEngineering: Treatment and Reuse." New York, NY, USA: McGraw-Hill, 2003.

[131] J. Pośpiech, "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), Aug. 2021, pp. 145–149. doi: 10.1109/MMAR49549.2021.9528445.

[132] M. Michałkiewicz et al., Poradnik eksploatatora oczyszczalni ścieków. PZITS Poznań, wydanie 3, rozszerzone, 1150 s. 2011.

[133] J. Pośpiech, W. Nocoń, and K. Stebel, "Agent-based system for continuous control and its application to activated sludge process," Bulletin of the Polish Academy of Sciences. Technical Sciences, vol. 72, no. 4, 2024, doi: 10.24425/bpasts.2024.150114.

[134] M. Czyżniewski, R. Łangowski, and R. Piotrowski, "Respiration rate estimation using non-linear observers in application to wastewater treatment plant," Journal of Process Control, vol. 124, pp. 70–82, Apr. 2023, doi: 10.1016/j.jprocont.2023.02.008.

[135] B. Carlsson, C.-F. Lindberg, S. Hasselblad, and S. Xu, "On-Line Estimation of the Respiration Rate and the Oxygen Transfer Rate at Kungsangen Wastewater Treatment Plant in Uppsala," Water Science and Technology; London, vol. 30, no. 4, pp. 255–263, Aug. 1994.

[136] T. Suchodolski, M. A. Brdys, and R. Piotrowski, "RESPIRATION RATE ESTIMATION FOR MODEL PREDICTIVE CONTROL OF DISSOLVED OXYGEN IN WASTEWATER TREATMENT PLANT," IFAC Proceedings Volumes, vol. 40, no. 9, pp. 286–291, Jan. 2007, doi: 10.3182/20070723-3-PL-2917.00046.

[137] M. A. Brdys, W. Chotkowski, K. Duzinkiewicz, K. Konarczak, and R. Piotrowski, "TWO - LEVEL DISSOLVED OXYGEN CONTROL FOR ACTIVATED SLUDGE PROCESSES," IFAC Proceedings Volumes, vol. 35, no. 1, pp. 467–472, Jan. 2002, doi: 10.3182/20020721-6-ES-1901.01387.

[138] A. C. Badino, M. C. R. Facciotti, and W. Schmidell, "Improving k La determination in fungal fermentation, taking into account electrode response time," Journal of Chemical Technology & Biotechnology, vol. 75, no. 6, pp. 469–474, 2000, doi: 10.1002/1097-4660(200006)75:6<469::AID-JCTB236>3.0.CO;2-4.

[139] S. Gerksic, D. Vrecko, and N. Hvala, "Improving oxygen concentration control in activated sludge process with estimation of respiration and scheduling control," Water science and technology : a journal of the International Association on Water Pollution Research, vol. 53, pp. 283–91, Feb. 2006, doi: 10.2166/wst.2006.133.

[140] J. Krampe and K. Krauth, "Oxygen transfer into activated sludge with high MLSS concentrations," Water Sci Technol, vol. 47, no. 11, pp. 297–303, Jun. 2003, doi: 10.2166/wst.2003.0618.

[141] S. A. A. Rawoof et al., "Production of optically pure lactic acid by microbial fermentation: a review," Environ Chem Lett, vol. 19, no. 1, pp. 539–556, Feb. 2021, doi: 10.1007/s10311-020-01083-w.

[142] FIPA Agent Management Specification http://www.fipa.org/specs/fipa00023/XC00023H.html

[143] FIPA ACL Message Structure Specification http://www.fipa.org/specs/fipa00061/SC00061G.html

[144] J. Pośpiech, "Rules for efficient development of multiagent systems for continuous process control," ELECTROTECHNICAL REVIEW, vol. 1, no. 8, pp. 227–231, Aug. 2024, doi: 10.15199/48.2024.08.46.

[145] 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.

[146] 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.

[147] M. Uschold and M. Gruninger, "Ontologies: principles, methods and applications," The Knowledge Engineering Review, vol. 11, no. 2, pp. 93–136, Jun. 1996, doi: 10.1017/S0269888900007797.

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

[149] MQTT Specification https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[150] M. Hadzic, E. Chang, and P. Wongthongtham, Ontology-Based Multi-Agent Systems. Springer Publishing Company, Incorporated, 2014.

[151] N. Noy and D. Mcguinness, "Ontology Development 101: A Guide to Creating Your First Ontology," Knowledge Systems Laboratory, vol. 32, Jan. 2001.

[152] D. Choinski and M. Senik, "Ontology Based Knowledge Management and Learning in Multi-Agent System," in Agent and Multi-Agent Systems. Technologies and Applications, G. Jezic, M. Kusek, N.-T. Nguyen, R. J. Howlett, and L. C. Jain, Eds., Berlin, Heidelberg: Springer, 2012, pp. 65–74. doi: 10.1007/978-3-642-30947-2_10.

[153] C.-J. Su and C. W. Peng, "Multi-agent ontology-based Web 2.0 platform for medical rehabilitation," Expert Systems with Applications, vol. 39, no. 12, pp. 10311–10323, Sep. 2012, doi: 10.1016/j.eswa.2011.09.089.

[154] M.-H. Wang, C.-S. Lee, K.-L. Hsieh, C.-Y. Hsu, G. Acampora, and C.-C. Chang, "Ontology-based multi-agents for intelligent healthcare applications," J Ambient Intell Human Comput, vol. 1, no. 2, pp. 111–131, Jun. 2010, doi: 10.1007/s12652-010-0011-5.

[155] M. Hadzic and E. Chang, "Ontology-based Multi-agent Systems Support Human Disease Study and Control," in Self-Organization and Autonomic Informatics (I), IOS Press, 2005, pp. 129–141. Accessed: Jun. 25, 2025. [Online]. Available: https://ebooks.iospress.nl/volumearticle/1796

[156] T. Olha, T. Kostiantyn, and T. Oleksandr, "Designing Intelligent Multi-agent Ontology-Based Training Systems: The Case of State University of Infrastructure and Technology," in Advances in Computer Science for Engineering and Manufacturing, Z. Hu, S. Petoukhov, F. Yanovsky, and M. He, Eds., Cham: Springer International Publishing, 2022, pp. 181–192. doi: 10.1007/978-3-031-03877-8_16.

[157] Z. Samia, R. Khaled, and Z. Warda, "Multi-Agent Systems and Ontology for Supporting Management System in Smart School," in 2018 3rd International Conference on Pattern Analysis and Intelligent Systems (PAIS), Oct. 2018, pp. 1–8. doi: 10.1109/PAIS.2018.8598505.

[158] B. L. Sadigh, Unver ,Hakki Ozgur, Nikghadam ,Shahrzad, Dogdu ,Erdogan, Ozbayoglu ,A. Murat, and S. E. and Kilic, "An ontology-based multi-agent virtual enterprise system (OMAVE): part 1: domain modelling and rule management," International Journal of Computer Integrated Manufacturing, vol. 30, no. 2–3, pp. 320–343, Mar. 2017, doi: 10.1080/0951192X.2016.1145811.

[159] D. Lavbič, Vasilecas ,Olegas, and R. and Rupnik, "Ontology-based multi-agent system to support business users and management," Ukio Technologinis ir Ekonominis Vystymas, vol. 16, no. 2, pp. 327–347, Jan. 2010, doi: 10.3846/tede.2010.21.

[160] G. Bella, D. Cantone, C. F. Longo, M. Nicolosi-Asmundo, and D. F. Santamaria, “The ontology for agents, systems and integration of services: OASIS version 2,” Intelligenza Artificiale, vol. 17, no. 1, pp. 51–62, Jun. 2023, doi: 10.3233/IA-230002

[161] K. Tkachenko and O. Tkachenko, “Modeling and using intelligent multi-agent system in smart city: ontological approach,” Transport systems and technologies, no. 42, pp. 45–57, Dec. 2023, doi: 10.32703/2617-9059-2023-42-4.

[162] N. Anand, van Duin ,Ron, and L. and Tavasszy, “Ontology-based multi-agent system for urban freight transportation,” International Journal of Urban Sciences, vol. 18, no. 2, pp. 133–153, May 2014, doi: 10.1080/12265934.2014.920696.

[163] M. Y. Aghdam, S. R. K. Tabbakh, and S. J. M. Chabok, “A New Ontology-Based Multi-Agent System Model for Air Traffic Management,” International Journal of Transportation Engineering vol. 10, 2022.

[164] S. Mandujano, A. Galvan, and J. A. Nolazco, “An ontology-based multiagent approach to outbound intrusion detection,” in The 3rd ACS/IEEE International Conference onComputer Systems and Applications, 2005., Jan. 2005, pp. 94-. doi: 10.1109/AICCSA.2005.1387085.

[165] P. Wongthongtham, E. Chang, and T. S. Dillon, “Ontology-based multi-agent system to multi-site software development,” in Proceedings of the 2004 workshop on Quantitative techniques for software agile process, in QUTE-SWAP ’04. New York, NY, USA: Association for Computing Machinery, Nov. 2004, pp. 66–75. doi: 10.1145/1151433.1151443.

[166] J. L. Hippolyte et al., “Ontology-based demand-side flexibility management in smart grids using a multi-agent system,” in 2016 IEEE International Smart Cities Conference (ISC2), Sep. 2016, pp. 1–7. doi: 10.1109/ISC2.2016.7580828.

[167] G. Santos, F. Silva, B. Teixeira, Z. Vale, and T. Pinto, “Power Systems Simulation Using Ontologies to Enable the Interoperability of Multi-Agent Systems,” in 2018 Power Systems Computation Conference (PSCC), Jun. 2018, pp. 1–7. doi: 10.23919/PSCC.2018.8442888.

[168] J. Du, H. Jing, K.-K. R. Choo, V. Sugumaran, and D. Castro-Lacouture, “An Ontology and Multi-Agent Based Decision Support Framework for Prefabricated Component Supply Chain,” Inf Syst Front, vol. 22, no. 6, pp. 1467–1485, Dec. 2020, doi: 10.1007/s10796-019-09941-x.

[169] G. Wang, T. N. Wong, and X. Wang, “An ontology based approach to organize multi-agent assisted supply chain negotiations,” Computers & Industrial Engineering, vol. 65, no. 1, pp. 2–15, May 2013, doi: 10.1016/j.cie.2012.06.018.

[170] P. Skobelev, A. Zhilyaev, V. Larukhin, S. Grachev, and E. Simonova, “Ontology-based Open Multi-agent Systems for Adaptive Resource Management,” presented at the 12th International Conference on Agents and Artificial Intelligence, Jun. 2025, pp. 127–135. Accessed: Jun. 22, 2025. [Online]. Available: https://www.scitepress.org/Link.aspx?doi=10.5220/0008896301270135

[171] J. Lim, L. Pfeiffer, F. Ocker, B. Vogel-Heuser, and I. Kovalenko, “Ontology-Based Feedback to Improve Runtime Control for Multi-Agent Manufacturing Systems,” in 2023 IEEE 19th International Conference on Automation Science and Engineering (CASE), Aug. 2023, pp. 1–7. doi: 10.1109/CASE56687.2023.10260621.

[172] D. Choinski and M. Senik, “Collaborative Control of Hierarchical System Based on JADE,” in Cooperative Design, Visualization, and Engineering, Y. Luo, Ed., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2010, pp. 262–269. doi: 10.1007/978-3-642-16066-0_39.

[173] D. Choiński, W. Nocoń, and M. Metzger, “Multi-Agent System for Collaboration in Hybrid Control,” in Computational Science – ICCS 2008, M. Bubak, G. D. van Albada, J. Dongarra, and P. M. A. Sloot, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 381–388. doi: 10.1007/978-3-540-69389-5_44.

[174] Ontoligua - http://www.ksl.stanford.edu/software/ontolingua/

[175] DAML - https://www.daml.org/ontologies/

[176] D. N. Batanov and W. Vongdoiwang, “Using Ontologies to Create Object Model for Object-Oriented Software Engineering,” in Ontologies: A Handbook of Principles, Concepts and Applications in Information Systems, R. Sharman, R. Kishore, and R. Ramesh, Eds., Boston, MA: Springer US, 2007, pp. 461–487. doi: 10.1007/978-0-387-37022-4_16.

Index of abbreviations

ACL	Agent Communication Language
BBPC	Boundary-Based Predictive Controller
CV	Control Value
DF	Directory Facilitator
DO	Dissolved Oxygen
FIPA	Foundation for Intelligent Physical Agents
FOPDT	First-Order Plus Dead Time
GUI	Graphical User Interface
IAE	Integral Absolute Error
IEEE	Institute of Electrical and Electronics Engineers
IOT	Internet Of Things
ISE	Integral Squared Error
JADE	Java Agent Development Framework
JSON	JavaScript Object Notation
LA	Lactic Acid
LLM	Large Language Model
MAS	Multi-Agent System
MIMO	Multiple Input, Multiple Output
OPC UA	Open Platform Communication Unified Architecture
OUR	Oxygen Uptake Rate
PC	Personal Computer
PLA	Poly-Lactic Acid
PLC	Programmable Logic Controller
PWM	Pulse Width Modulation
RAC	Robust Adaptive Controller
SCADA	Supervisory Control And Data Acquisition
SP	Set Point
UAV	Unmanned Air Vehicle
UML	Unified Modeling Language
WWTP	Wastewater Treatment Plant
XML	Extensible Markup Language

List of figures

List of tables