16

Conscious Agents Interaction Framework for Industrial Automation

Polina Ovsiannikova[1] and Valeriy Vyatkin[1,2]

[1] Aalto University, Finland
[2] Luleå Tekniska Universitet, Sweden

Abstract

This work-in-progress paper explores the integration of human cognition and interaction models into industrial automation systems. We begin by examining how human cognitive patterns can be applied to the development of conscious agents within these systems. From an interaction standpoint, we analyse the dual role of humans as both users and workers within automated environments. The convergence of these perspectives enables the creation of intelligent, multi-agent systems where humans function as equal agents. Such systems are characterised by true flexibility, as each component can independently assess its capabilities and collaboratively plan actions to achieve both collective (external) and individual (internal) goals. We present two case studies to illustrate these concepts: The first case study examines a distributed vertical farm, where modules must coordinate their energy consumption, demonstrating steps towards cognitive reasoning in machines. The second involves the automation of a cruise ship's HVAC system, where agents (cabin units) negotiate a temperature setpoint based on human behaviour (user-focused scenario).

Keywords: multiagent systems, human-computer interaction, computational rationality, reconfigurable manufacturing systems.

16.1 Introduction

Flexibility and reconfigurability are essential for driving automation processes based on demand. These characteristics are applicable across a range of domains, including building automation (e.g., HVAC systems, lighting control), pharmaceutical production, smart factories, and other manufacturing systems.

Reconfigurability is also critical in situations such as changing energy consumption targets, emergency scenarios, equipment upgrades, or the introduction of new workers who need time to reach full capacity while production targets remain unchanged. Similarly, adapting to external factors, such as increased HVAC demands—ensures that system goals are met without compromising productivity, even when resources are limited.

The dynamic requirements in these contexts arise from both internal and external factors. The internal ones are related to business processes and goals, liveness, and safety properties of the systems concerning with respect to their momentary conditions. External, in turn, are often influenced by general economic and environmental conditions. These requirements can be introduced during runtime or can be pre-defined.

Internal and external requirements can conflict, even if initially they appear aligned. For example, a predefined internal goal of meeting the minimal production output might contradict a dynamic external demand to reduce energy consumption. One solution would be to set priorities among requirements, while another approach involves optimising the performance of the system to balance competing needs.

Pre-programming for such optimisation is possible, but as systems grow in complexity, maintaining them becomes increasingly effort-intensive. A more scalable solution involves creating multi-agent systems (MAS), where each agent "knows" its capabilities, can "perceive" and adapt to changing goals.

In this paper, we propose a MAS framework based on self-conscious intelligent agents motivated by human cognition and capable of perceiving humans as equals.

16.2 Related Research

The concept of multi-agent systems (MAS) is not new, nor is the idea of conscious machines (both can be found in research and sci-fi novels). MAS is a system that involves multiple intelligent agents that interact with each other

and solve a set of problems. In this paper we consider the industrial application of MAS, meaning that our agents are control programs for processes and devices in an industrial automation setting. Thus, for example, [1] discusses the use of agent technology in smart grid automation using the standards IEC 61499 and IEC 61850, [2] develops the idea of intelligent mechatronic components, and [3] uses an agent-based approach to model the dynamics of liquid goods logistics and energy efficient sensors usage.

While we refer the reader to [4] for the introduction of MAS, the works on the second concept we discuss here in more detail.

One way to bring consciousness to machines is to implement a belief-desire- intention (BDI) model [5]. In the BDI model, intentions are primarily perceived as elements of partial plans of action (although it is argued that BDI models do not include a built-in capacity for "lookahead" type of planning [6]). It reflects how humans reason and form intentions based on their goals and the information they have. In this model, belief represents the agent's perception of the environment, desire refers to the goals the agent wants to achieve (which can be organised hierarchically, although this is not always necessary), and intention is the agent's commitment to a specific course of action based on a plan.

The BDI model has its limitations, and a series of works tackles them. For example,

[7] complements the reasoning of BDI agents about time with reasoning in time, making the model suitable for dynamic environments. Then, the BDI model does not describe the communication between agents within the context of MAS, which is partially dealt with in [8] where LORA (the logic of Rational Agents) allows reasoning about interaction and incorporates action logic. An important extension of the BDI model is [9]. The author integrates a decision-theoretic planning framework into BDI and proposes the concept of bounded rationality. In this approach, the agent's decision-making is based on utility maximisation while also considering cognitive and environmental constraints, making it a more realistic model of rational behaviour under limitations.

Computational rationality [10] is based on the core ideas of bounded rationality. In addition to the mentioned constraints (cognitive, environmental, etc.), it considers the computational cost of the decision-making process. A solution to a computation rationality problem is optimal with respect to the trade-offs between the quality of decisions and the cost of computation. In [10], the authors highlight that in many cognitive science frameworks, decision-making is often modelled with two distinct levels, i.e., the higher

rational level which defines goals and strategies, and the lower mechanistic level which focuses on the implementation or on how the agent uses its cognitive mechanisms (e.g., perception, memory) to approximate or achieve these goals in practice. Computational rationality, in turn, treats information-processing limitations (finite memory, time, or computational resources) as integral parts of the model of rational behaviour. Unlike traditional models of rationality, which assume agents have unlimited cognitive capacity, computational rationality integrates computational and information-processing constraints directly into the definition of rational behaviour.

The framework offers a structured way to explore how behaviours arise from cognitive mechanisms that are tailored not just to the environment, but also to the cognitive structure of the mind. Within this system, optimal behaviour is understood as the result of executing the best possible program suited to a given environment. This behaviour represents the machine's upper limit of performance under the given conditions.

The primary focus of [10] is how individual agents make decisions, so, although in the context of MAS, other agents can be viewed as the environment of individual agents they perceive, there is no specific focus on the interaction between agents, including scenarios where a human is perceived as an agent. In [12], the authors, in turn, lay the concept of computational rationality as a foundation for a theory of interaction within the domain of human-computer interaction (HCI).

While modelling the interaction between a computer and a human (e.g., through a graphical user interface or other means), the modeller must guess human reactions and encode them into a set of rules. This is a nontrivial task as users often deviate from the standard behaviour, which requires machines to be flexible in their replies.

In the industrial automation world, the problem is especially acute as the mental states of both human workers and human users are subject to change. Depending on the length of the collaborative production scenario, it can further alter and require a rapid, safe, and performance-optimal response from the machine. While [12] does not tackle industrial automation scenarios per se, it argues that computationally rational agents can successfully interact with humans.

The core idea is that interactive behaviour results from a control policy that is optimally adapted to users' preferences and constraints. These constraints arise both from the internal environment (cognitive limits) and the external environment (the interaction context). Interaction, therefore, is seen

as the rational outcome of these bounded choices. By utilising approximate optimization techniques like reinforcement learning (RL), hypotheses about user goals and processing limits can be generated, allowing for adaptable strategies. The theory also allows the machines to generate explanations of human actions, i.e., answer "why?" and "what if?" questions, which allows them to be more adaptive. The need for adaptiveness is also motivated by the fact that individuals often distribute cognitive processing across their environment to optimise the use of their mental and cognitive resources, this is known as the adaptive distribution of cognition [13].

The works discussed provide a foundation for exploring computational rationality as a framework for agent interaction. Notably, we do not differentiate between human and machine agents in our approach. This leads us to our central question: "If we develop intelligent agents that approximate human cognition, can we integrate humans into the system as equal agents?" Our research objective is to create a multi-agent system (MAS) interaction framework that allows agents to be dynamically added or removed while pursuing both individual and collective system goals. This framework should have adaptable communication strategies to prevent using inefficient methods of problem-solving (e.g., becoming trapped in local optima [13]) when evidence suggests a better approach exists.

The central point in the agent architecture we envision is a self-awareness layer (or if we follow terminology from [12], the internal environment). Suppose we aim to avoid hard-coding production recipes or human-machine interaction. In that case, the control program should be able to "reflect" on itself and "decide" how its capabilities and limitations fit into the current set of system goals. For instance, consider an agent which is a program that controls a robotic arm. It can make the arm move and perform grip physically, but if it lacks awareness of these capabilities and is asked to retrieve a book from a shelf, the agent might respond that it cannot perform the task. One might argue that a self-awareness layer can be easily generated since an agent is a control program with the code often known.

Surely, if the agent is a white box and the control program is deterministic, one can very well predict its future behaviour knowing, for example, its execution traces. However, this luck is rare, and agents can be represented by a neural network or other machine learning models.

The next point is that to achieve the full coverage of the environment relevant to the agent, it needs to learn not only its own capabilities but also how they fit into the whole system and, hence, what other agents do. Thus,

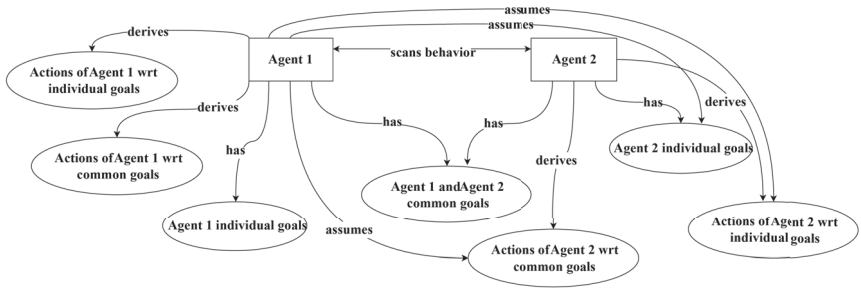

Figure 16.1 Ontology showing the interaction of two agents. Assume-Derive phases are shown for Agent 1.

they need to observe the behaviour of other agents, assume what they could do with respect to the current goals, and derive their behaviour based on this knowledge and their goals. The assume-derive model is shown in Figure 16.1.

16.3 Interaction Framework

In this section, we outline the framework within which we can achieve the integration of human cognition principles into the MAS system in a way that a human is treated as an equal agent. It consists of four layers: automation, self- awareness, high-level agent communication, and the highest level where the user specifies the system goals. Each layer at any time must ensure safe operation and pass formal checks when necessary. For each of the layers, in addition to their description, we also name the major problems to be solved.

16.3.1 Layer 1: Automation

We work with industrial automation systems; thus, our first layer is the layer of pure automation control programs. We can operate out of the idea that our low-level control programs are smart and can rewrite their sequence of actions, and such work is being carried out. However, at this stage, we assume that they operate with a set of atomic operations that can be activated in any order by a scheduler from the higher level (e.g., using OPC UA). An atomic operation has a common definition and different specifics depending on the implementation of the automation system. Commonly, an atomic operation is a single action performed by a machine that starts in the safe state of the system and leaves the system in a safe state ready for next allowed actions.

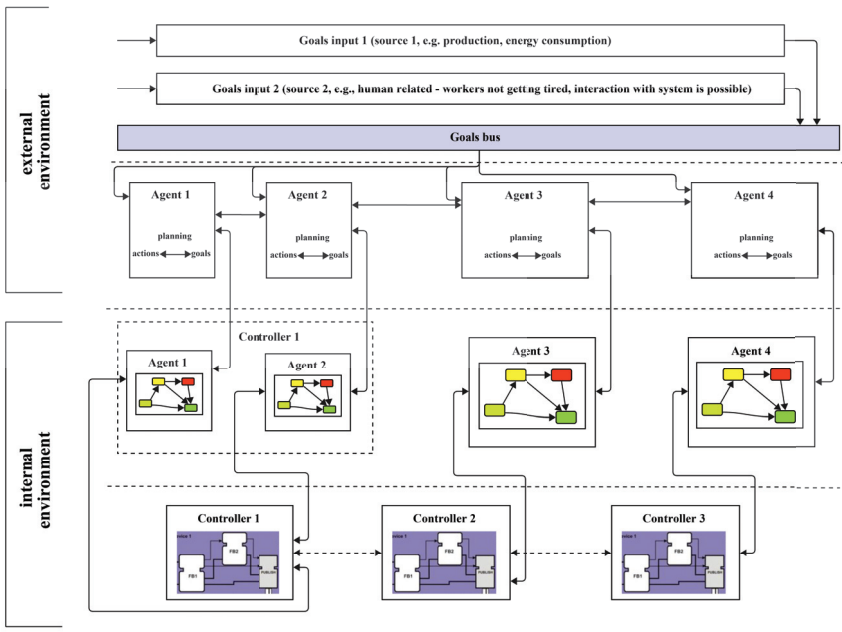

Figure 16.2 Interaction framework.

Consider having a drilling machine with a rotating table. Atomic operation "perform drilling" in addition to lowering the drill machine and turning it on for a defined amount of time, will also include checking if there is a workpiece underneath it. The same suffices with any other process constraints; for example, if there is a constraint on the maximum water temperature in the tank, the operation "heat" will not turn on the heater if the temperature has reached its maximum level.

An automation system can be represented as PLC control code with a human-machine interface (HMI) and in this case, we must interface the automation program directly, e.g., by sending data and events to function blocks, in case it is implemented in IEC 61499. Another option is to send the sequences of actions to the manufacturing execution system (MES), which has the set of possible actions defined. Thus, this level directly executes the commands from the upper layer and reports on the results of the operations performed back to the top. The PLC code can also be organised in a way that supports the MAS paradigm on this level to ease the debugging process.

16.3.2 Layer 2: Self-awareness

Self-awareness is one of the core layers of the framework, where the concept of an agent is introduced. As highlighted in Figure 16.2, the automation layer only implements control and equipment-specific safety constraints. The self-awareness layer, in turn, defines which automation operations can be grouped into a wholesome entity together with environmental observations specific to the operations. The grouping can be obvious, e.g., a conveyor belt or a robot arm can be self-sufficient agents. On the other hand, if the robot arm is installed on AGV, a composite agent can represent the whole assembly. Another way to define agents is by following the control loops of the system. Thus, having a vertical farming module with lighting and watering systems, the lighting agent would be capable of adjusting the light according to the lighting schedule and, for example, the PAR (Photosynthetically Active Radiation) sensor. The watering agent then is responsible for the regulation of the water flow depending on the watering schedule and a water flow sensor that detects clogs in the pipes.

The self-awareness layer can be developed through RL methods. For example, using data from factory processes, an offline RL method can learn the system's dynamics and generate an optimal control policy based on past experiences. In this approach, agents are trained to become aware of their capacities through iterative learning and feedback [14].

If a digital twin or a simulation model of the process is available, such development can be supported by a closed-loop system, comprising two main components: a modelling toolbox and a controlling toolbox. In [15], the modelling toolbox allows the creation of a plant model that simulates various system components or the entire system, which, in our case, would transform into a simulation of the controller actions and environmental response. Meanwhile, the controlling toolbox, originally, is a collection of autonomous agents that interact with the plant model using reinforcement learning algorithms, which would be replaced by the "consciousness" of the agents being trained.

The main function of this layer is to provide specific to the request available abilities and skills of the agent to the upper layer with the data needed for scheduling and planning and to translate these results into a sequence of atomic operations. The definition of appropriate reward functions for training the self-awareness layer to choose suitable capabilities is then necessary to keep the agents from "overthinking".

One interesting challenge in self-awareness is the possibility of discovering skills that were not demonstrated in the self-learning process. Assume, we have a conveyor belt, which is controlled by the motors and set to move workpieces with some fixed speed. Can the speed be changed to achieve production or energy consumption system goals? Whether the agent can discover new skills safely during the runtime is one of the research questions related to this layer.

Another challenge, stemming from the fact that the self-awareness of our agents is remote from their physical abilities, is how the self-awareness layer can adjust to the change of equipment executing approximately the same skills and actions. The advantage of our framework is that the upper layers do not have to be aware of such changes, while the self-awareness layer will have to perform self-discovery.

16.3.3 Layer 3: High-level communication and coordination

In this layer, agents communicate and coordinate the completion of tasks to achieve the goals passed down from the upper layer, thus we outline the following main functions of the level:

Goal selection. On the top level, the goals can be determined for the whole system (e.g., total energy consumption must be X) or for the subsystems (e.g., the human in the packaging unit should be protected from burning out). Agents must be able to select which goals are theirs to complete and which are for the other agents. Goals can also be adopted from other agents when they determine that assistance is needed.

Agents' collaboration. Agents should be able to group (or "team up") to collaboratively achieve the goals if needed. For example, for a human not to get a burn-out in the packaging unit, probably, a conveyor belt should work slower after lunch and an AGV should pick the packed products faster to keep the workspace of the human clean. For this, agents (1) must assume the goals of other agents, as precisely as they can, (2) assume their common and individual goals, and (3) derive their action based on the requirement inferred using this information. We call these two phases the "Assume" phase and the "Derive" phase (Figure 16.1). Reasoning can help with the Assume phase, where an agent can analyse the trace of decisions of other agents and find their motivation (for example, in [16]). This will allow agents to plan their actions in the context of plans of other agents.

Dynamic optimal policy determination. Some agents might have several goals to satisfy, which, in turn, can be contradictory and belong to different groups of agents. An agent must be able to find an optimal course of action. A crucial part of this decision-making process is evaluating utility [12]. The utility of an action is not limited to the immediate reward it brings but also depends on the future rewards that can be expected when the same policy is followed over time. By considering both short-term and long-term outcomes, agents can make decisions that optimise their overall performance in achieving individual and system-wide objectives.

External environment update handling. Agents can be sprouted and destroyed dynamically, depending on the momentary requirements, which they should be able to communicate and, thus, perceive when they are notified about the same events from the other agents. The reconfiguration in this case must happen automatically with as little downtime or human intervention as possible.

Communication with humans. Agents that interact with humans should have the mechanisms to assert the same aspects of the human internal environment as they do when communicating with nonhuman agents.

In the implementation of the framework for this layer, we also consider a time aspect of the decision-making. If finding an optimal action plan takes a significant number of resources (time, computational power, etc.), the heuristics are to be used. Also, here, we solve the problem of the cost of changing the plan if a new better option is found. For instance, if midway through the plan execution, the resources are already spent, this layer estimates whether replanning is beneficial.

16.3.4 Layer 4: System goals

Mainly the goals sprout from the requirements for the system, about which we talked in Section I. Different formulation languages are possible, for example, various temporal logics, diagrams, or natural language, which is becoming more possible with the development of LLMs. This layer is also the place for enlarging the vocabulary of the system, for example, if the goal is "to keep power consumption at 80% from maximum consumption during the nearest month", the system should at least know what power consumption is. LLMs and users play a key role in vocabulary and semantics enrichment.

The requirements the system should satisfy may have a time interval, for example, different production scenarios of the island-based factory might

follow different production recipes, however, the general liveness and safety requirements for the production may remain. One important note is that we do not differentiate here goals for machines and goals for humans within the production. Human workers or users participate in goal distribution on the same level as nonhuman agents do. The difference shows only in the way the goals are communicated to a human as special interfaces are required for the purpose.

16.4 Case Study

As a leitmotif for the framework trials, we chose the topic of energy consumption, since, in large productions, being able to reduce it even for a small fraction of the total leads to a significant impact on sustainability and production costs. Our case studies explore different parts of the framework implementation starting from communication between the control loops of the assembly, where each of them can straightaway decide on energy consumption to indirect interaction with humans and agents that influence the energy consumption of the main consumers but not decide on it straight away.

16.4.1 Vertical farming module

Our first case study is on the optimisation of the energy consumption of a vertical farming module. Figure 16.3 shows the prototype module implemented at the Aalto Factory of the Future and its overall architecture. Our automation code runs on two M251 PLCs to which both digital and analogue sensors and actuators connect. This layer connects to the business logic via OPC UA. The

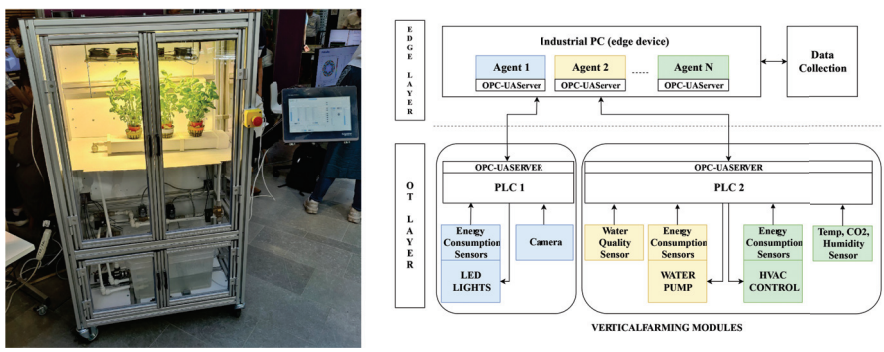

Figure 16.3 Vertical farming module and its controller architecture.

business logic layer is placed on the industrial PC, while it is also possible to develop it in the cloud.

Our high-level logic employs the MAS concept. Each agent here is responsible for calculating the consumption of a single control loop, where a control loop is composed of an actuator and the environmental response on the action of the actuator, for example, pump–water level. The whole module itself can also be an agent in case there is more than one. In this case, it becomes a composed agent, incorporating a set of agents—control loops. Agents communicate with each other to converge on the best individual power consumption values, given the total target power consumption of the system.

In this example, we explicitly specified the self-conscious layer by inputting the characteristics of the actuators controlled during the experiments, namely LED lights of the vertical farming module. We have also experimented with setting up communication between different agents through distributed optimisation methods (for details, see [16]).

16.4.2 HVAC system of a cruise ship

Another case study we have in progress is the energy consumption of a cruise ship. Here, several cabins connect to one Air Handling Unit (AHU) and several AHUs connect to a diesel engine. In HVAC, the parts that consume the most energy are the fans that drive the air inside the main ducts and the chiller that cools it down to a set temperature. The only way we can influence the energy consumption decision is by adjusting the temperature set points in the cabins. Thus, lowering the cabin temperature would mean that the corresponding valve for fresh air intake should open more, which decreases the pressure in the main air duct, making the main fan increase power consumption to stabilise the pressure, which also makes the chillers cool down a larger air volume.

Since the solution should be flexible (extendable to various actuator types) and scalable (more cabins can be connected to air ducts/more AHUs, diesel engines in the system), we define each cabin, each AHU, and each diesel as an agent. This relieves us from the need to explicitly encode the physics of the process (unlike in the first case study) and AHU "learns" its dynamics of power consumption based on the traces of the simulation model using RL methods. One of the research questions here is how to dynamically generate not only the agents but also the correct way of communication between them.

Another angle of this study is incorporating users' patience into the system. The energy reserves are restricted on the cruise ship. Suppose that a lot of people set the desired temperature below some minimum. In that case, there is a chance that the diesel engines will not be able (within safety boundaries) to provide enough energy for the ship infrastructure and the HVAC system. This means that the temperature set point should be adjusted automatically; however, this works up to a point where people do not massively complain.

16.5 Discussion and Conclusion

While extensive research has been conducted on isolated issues in MAS and HCI, little attention has been paid to the design of intelligent systems as a whole, especially, from the industrial automation perspective. This paper outlines the foundational problems and research questions aimed at shaping machine consciousness, enabling agents to understand one another as well as communicate effectively with humans.

We hypothesise that the implementation of such an architecture leads to emergent behaviours akin to needs and motivations for the system itself. As individual agents become more aware and coordinated, the factory's operations could form a collective consciousness, transforming the production process into something that responds naturally to the environment rather than following pre-set instructions. This represents the basis of a truly reconfigurable and flexible factory—one where manufacturing as a service becomes an inherent response to the factory's needs, rather than an artificially imposed command structure.

Future work will involve a deeper analysis of the solutions proposed for each layer of the system, with special emphasis on the communication aspects, both within the machine agents and between agents and human workers, which involves exploring how intelligent agents can effectively communicate and collaborate with humans.

Furthermore, experiments will be conducted in environments with static and dynamic actuators, including mobile workstations, AGVs, and reconfigurable factories. This framework will also have applications in energy communities, where negotiation and flexibility with human input is vital, especially for systems without plug-and-play capabilities, like HVAC.

Acknowledgements

This work was supported, in part, by projects I-SWARM X (p/n 411061), Multi- Energy VPP (p/n 13348415), and the project funded by the Academy of Finland, CAFE (p/n 13363691).

References

[1] G. Zhabelova, V. Vyatkin, and V. N. Dubinin, "Toward Industrially Usable Agent Technology for Smart Grid Automation," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 4, pp. 2629–2641, Apr. 2015,

[2] A. Kalachev, G. Zhabelova, V. Vyatkin, D. Jarvis and C. Pang, "Intelligent Mechatronic System with Decentralised Control and Multi-Agent Planning," *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society*, Washington, DC, USA, 2018, pp. 3126-3133,

[3] J. Kaiser, M. P. Hernández, V. Kaupe, P. Kurrek, and D. McFarlane, "An agent-based approach for energy-efficient sensor networks in logistics," *Engineering Applications of Artificial Intelligence*, vol. 127, p. 107198, Jan. 2024,

[4] M. J. Wooldridge, "An introduction to multiagent systems". Chichester: John Wiley, 2012,

[5] M. E. Bratman, D. J. Israel, and M. E. Pollack, "Plans and resource-bounded practical reasoning," *Computational Intelligence*, vol. 4, no. 3, pp. 349– 355, Sep. 1988,

[6] S. Sardiña, L. de Silva, and L. Padgham, "Hierarchical planning in BDI agent programming languages," *5th international joint conference on Autonomous agents and multiagent systems (AAMAS '06)*, Association for Computing Machinery, New York, NY, USA, pp. 1001–1008, May 2006,

[7] F. Alzetta, P. Giorgini, M. Marinoni, and D. Calvaresi, "RT-BDI: A Real-Time BDI Model," *Advances in Practical Applications of Agents, Multi-Agent Systems, and Trustworthiness*, The PAAMS Collection, pp. 16–29, 2020,

[8] M. Wooldridge, "Reasoning about Rational Agents." *MIT Press*, 2003,

[9] G. Boella, "Decision theoretic planning and the bounded rationality of BDI agents," *In Proceedings of GTDT 2002 Workshop*, Technical report WS- 02, vol. 6, pp. 1-10, 2002,

[10] R. L. Lewis, A. Howes, and S. Singh, "Computational Rationality: Linking Mechanism and Behavior Through Bounded Utility Maximization," *Topics in Cognitive Science*, vol. 6, no. 2, pp. 279–311, Mar. 2014,

[11] A. Oulasvirta, J. Jokinen, and A. Howes, "Computational Rationality as a Theory of Interaction," *CHI Conference on Human Factors in Computing Systems (CHI '22),* Association for Computing Machinery, New York, NY, USA, Article 359, 1–14, 2022,

[12] S. J. Payne, A. Howes, and W. R. Reader, "Adaptively distributing cognition: A decision-making perspective on human - computer interaction," *Behaviour & Information Technology*, vol. 20, no. 5, pp. 339– 346, Jan. 2001,

[13] J. Deng, S. Sierla, J. Sun, and V. Vyatkin, "Offline reinforcement learning for industrial process control: A case study from steel industry," *Information Sciences*, vol. 632, pp. 221–231, Jun. 2023,

[14] Y. Berezovskaya, C.-W. Yang, and V. Vyatkin, "Reinforcement learning approach to implementation of individual controllers in data centre control system," *IEEE 20th International Conference on Industrial Informatics (INDIN)*, pp. 41–46, Jul. 2022,

[15] P. Ovsiannikova, I. Buzhinsky, A. Pakonen, and V. Vyatkin, "Oeritte: User- Friendly Counterexample Explanation for Model Checking," *IEEE Access*, vol. 9, pp. 61383–61397, 2021,

[16] K. Zhukovskii et al., "Energy Consumption Optimisation for Horticultural Facilities", *IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sep. 2024.