Supporting corporate AI awareness in remote teams through multi-agent systems

Matteo Manca[1,†], Stefano Tedeschi[2,*,†] and Cristina Baroglio[1,†]

[1] *Università degli Studi di Torino, Torino, Italy*
[2] *Università della Valle d'Aosta - Université de la Vallée d'Aoste, Aosta, Italy*

Abstract

The Covid-19 pandemic acted as an accelerator in the digitalization process of small-medium enterprises and public administrations, promoting new forms of hybrid/remote work. At the same time, companies and organizations increasingly rely on Artificial Intelligence (AI) technologies to streamline operations, promote innovation, and boost productivity. These deep changes call for the crucial deployment of new processes and adequate infrastructures to promote AI awareness among remote workers that may be geographically distributed. In this paper, we explore the role of multi-agent systems (MAS) in enhancing AI awareness and adoption in remote corporate teams. We sketch the architecture of a multi-agent system specifically designed to support the realization of dedicate AI training processes in corporate environments. The presented system leverages the SARL multi-agent programming framework.

Keywords

Multi-Agent Systems, Corporate AI training, Remote work, SARL

1. Introduction

In recent years, Artificial Intelligence (AI) adoption is experiencing exponential growth across companies, driven by the need to enhance operational efficiency, optimize decision-making processes, and innovate business models. Recent reports, such as those by PwC and McKinsey, highlight that almost 50% of organizations plan to integrate or have already integrated at least one AI technology into their workflows, with sectors ranging from healthcare to finance and manufacturing, leading the way in automation, predictive analytics, and AI-driven customer interaction platforms. However, despite this widespread deployment of AI, significant gaps in employee AI literacy and organizational preparedness remain, presenting critical challenges for AI integration success [1, 2].

In addition, the literature points out a disparity between the pace of AI adoption and the degree of awareness and understanding among the corresponding workforce. For instance, a study by Ransbotham et al., 2017, in the MIT Sloan Management Review highlights that while organizations are keen on adopting AI technologies, employees often lack clarity regarding AI's implications for their roles and career development [3]. This gap underscores the importance of targeted AI training programs that do more than just teaching the mechanics of AI systems; they must also foster a deeper comprehension of AI's potential to augment human capabilities rather than replace them.

In the context of a rapidly evolving work environment, complemented by the shift to remote and hybrid models of work following the Covid-19 pandemic, the challenge of equipping employees with AI skills becomes more and more complex. In this perspective, a problem that every company must cyclically face is the training of its employees. Keeping employees updated on new techniques and technologies is of great importance in maintaining the company's competitiveness in the market.

1st Workshop on Education for Artificial Intelligence (edu4AI 2024, https://edu4ai.di.unito.it/), co-located with the 23rd International Conference of the Italian Association for Artificial Intelligence (AIxIA 2024). 26-28 November 2024, Bolzano, Italy

[*] Corresponding author.

[†] These authors contributed equally.

matteo.manca@edu.unito.it (M. Manca); s.tedeschi@univda.it (S. Tedeschi); cristina.baroglio@unito.it (C. Baroglio)
0000-0002-9861-390X (S. Tedeschi); 0000-0002-2070-0616 (C. Baroglio)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

Enhancing employees' skills also allows for better results in a shorter time, especially in a field like AI. Remote work creates an additional layer of difficulty in delivering effective corporate training, as organizations must now rely on digital tools to engage a geographically dispersed workforce. Traditional in-person training models are proving to be inadequate in addressing these challenges, as highlighted by studies like [4], which emphasizes the need for scalable, adaptive, and engaging remote learning solutions.

The literature on remote corporate training has increasingly focused on leveraging digital technologies to meet these new demands [5, 6, 7, 8]. Digital learning platforms, AI-driven personalized learning pathways, and virtual collaborative environments have emerged as promising approaches. However, challenges persist in ensuring sustained employee engagement, providing real-time feedback, and tailoring training to individual skill levels in a remote setting. In particular, maintaining the same level of interaction and support that in-person training offers is a major concern, leading to difficulties in engaging remote workers during corporate training sessions.

Addressing these multifaceted challenges requires innovative solutions. In this paper, we propose the use of multi-agent systems (MAS) [9] as a robust computational framework for enhancing AI awareness and adoption within remote corporate teams. MAS, offer a dynamic and scalable approach to facilitating AI training, as already discussed in [10] in the context of AI-driven corporate simulations. These systems can simulate real-world organizational dynamics, allowing for adaptive and interactive learning experiences that are customized to individual employees, regardless of their physical location. We refer to this approach as "AI for AI" since we propose the use of an AI technology itself (i.e., multi-agent systems) as a tool to promote AI education in corporate remote teams. By leveraging MAS, organizations can not only streamline AI training processes but also foster a deeper understanding of AI's role, thereby promoting higher employee engagement and smoother AI integration across remote teams.

More in detail, we present the prototype of a multi-agent system specifically designed to support the realization of dedicate AI training processes in corporate environments. The presented system leverages the SARL multi-agent programming framework [11]. After a short introduction to SARL (in Section 2), Section 3 illustrates the general architecture of the system. Section 4 presents a general overview of the conducted simulation, while Section 5 provides some insights concerning the concrete implementation of the agents. Conclusions end the paper.

2. Background: SARL

SARL [11] is an open-source, agent-oriented programming language designed with a focus on high extensibility and rooted in object-oriented principles. Its high-level abstractions simplify the management of concurrent processes and interactions between agents, while still maintaining the autonomy of each agent. This flexibility allows agent functionalities to be dynamically altered during runtime.

An *agent* in SARL is an autonomous entity possessing various *capacities*, which are realized through specific *skills*. These capacities define the actions an agent can perform, though they require implementation via skills, which can be added either by default or dynamically. Capacities in SARL can extend others, much like class inheritance in object-oriented programming languages like Java.

A key feature of SARL is the ability to compose agents into hierarchical structures, known as *holonic systems*, where agents are nested within one another. Each "holon" maintains autonomy while interacting with other holons to achieve individual or shared objectives, but embodying at the same time a recursive complex system.

Agents in SARL are event-driven, reacting to events emitted in specific *spaces* where they are registered. These *events* trigger dedicated *behaviors*, which are sequences of actions that agents can execute. Behaviors can be invoked either by the agent itself, in response to an event, or after a predetermined time. Each agent undergoes a lifecycle that includes reacting to an `Initialize` event upon creation and responding to a `Destroy` event to terminate. During its lifecycle, an agent can execute additional behaviors triggered by specific events or time-based conditions. However, due to agents' autonomy, one agent cannot directly terminate another.

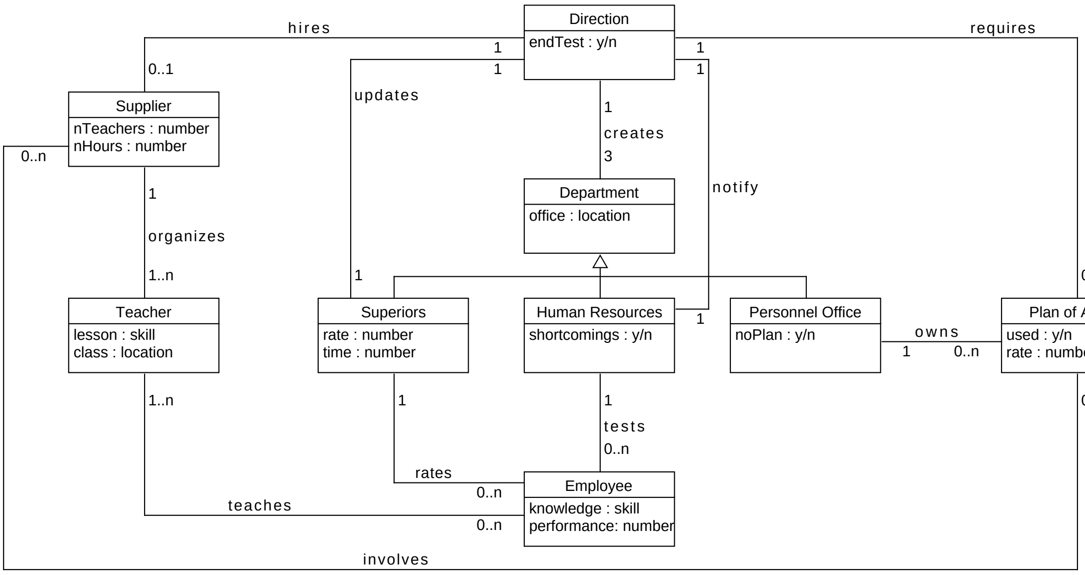

Figure 1: Domain model of the implemented multi-agent system.

Communication between agents, or even within the behaviors of a single agent, is primarily managed through events, as well. Events carry information such as their type, the agent that generated them, and specific attributes. Events are emitted into shared spaces, where they can be received or filtered by other agents without targeting any specific recipient.

This ability to model autonomous, interacting agents within distributed environments makes SARL particularly powerful for simulating complex systems, such as those needed for corporate training support systems. SARL's ability to represent dynamic, evolving processes aligns well with the our aim of modeling and tracking the learning patterns of employees in a corporate training context.

3. An agent-based architecture for corporate AI training

The multi-agent system developed in this work is designed to simulate a corporate AI training process, aiming to optimize the planning and execution of employee skill development programs in order to AI awareness and education among employees. In a dynamic business environment where continuous employee training is critical to maintaining competitive advantage, such a system may serve as a decision-support tool for companies. It allows management to simulate various training scenarios, assess the outcomes, and make data-driven decisions about the most efficient and cost-effective training paths. The main objective of the system is to minimize the cost and time required for training while ensuring that employees acquire the skills that are deemed as necessary.

According to [4], a training process is a complex workflow, which involves multiple stakeholders. First of all, the creation of a training plan is necessary to instruct employees. This plan must be based on a preliminary analysis of the missing skills of individual employees or groups. Such an analysis can be conducted periodically or following the definition of new company objectives by management. It is usually carried out by the supervisors of the involved employees, the heads of organizational functions, or through employee self-assessments. A form containing the training request is then completed and submitted to the personnel office, which will formalize it. It must then be authorized by the general manager to initiate the training process.

Figure 1 reports the general architecture of the implemented multi-agent system. SARL enables the creation of autonomous agents that represent the various entities involved in the training process. The system models key organizational components, such as departments, employees, and external training providers, as agents that interact dynamically within the simulated environment.

The core agents in the system include:

DirectionAgent This agent represents the management of the organization and acts as the coordinator of the simulation. It is responsible for initiating the simulation, overseeing the training process, and communicating with the other agents to execute the training plans.

EmployeeAgent Each employee agent simulates an individual employee with a specific skill set. These agents interact with training providers to acquire new skills, undergo periodic testing to assess skill development, and communicate with their respective departments regarding their training needs.

DepartmentAgent These agents represent organizational departments, each responsible for monitoring the training progress of employees. They communicate training requirements to employees and provide feedback to management on the effectiveness of the training programs. There are three types of department agent: (i) `HumanResources`, who tests the starting skills of employee agents; (ii) `PersonnelDepartment`, who provides the training plans; and (iii) `Superiors`, who tests the outcome of the current iteration of the simulation. These agents have the same structure but use different sets of skills based on the role assigned to them by the `DirectionAgent` at the beginning of the simulation and all have their own dedicated space for interaction.

SupplierAgent The supplier agent represents external training providers. It dynamically adapts its characteristics, such as course duration, cost, and reliability, based on the specific training plan provided by the management.

As said, the system follows an event-driven model, where agents communicate via events that trigger specific actions. For instance, once a training program is initiated by the `DirectionAgent`, `EmployeeAgents` are assigned to training sessions managed by `SupplierAgents`. `EmployeeAgents` then undergo skill development and are periodically tested by `DepartmentAgents`. Each agent operates autonomously, responding to the events it receives, which enables a flexible and reactive simulation of the corporate training process.

4. Simulation overview

The simulation has been designed to model the interaction between the various agents along the training process. It proceeds through a series of distinct phases, ensuring that agents are initialized, trained, evaluated, and eventually terminated. Below we briefly describe each step of the simulation process.

Initialization Phase The simulation begins with the creation of the root agent, `DirectionAgent`. This agent is responsible for orchestrating the overall flow of the simulation. The `DirectionAgent` spawns a specified number of `EmployeeAgent` instances, corresponding to the number of employees in the system. Each `EmployeeAgent` represents a distinct employee that will participate in the training simulation. Once all agents are spawned, `DirectionAgent` waits for the receipt of `AgentSpawned` events from all the `EmployeeAgent` instances. This synchronization ensures that the simulation starts only when all agents are successfully created and ready to interact.

Human Resources Phase The `DirectionAgent` triggers a `NofEmployees` event in the Human Resources space. This event signals the number of employees participating in the current iteration. The `DirectionAgent` registers itself as a “strong participant” in this space, ensuring that the Human Resources space remains active. Simultaneously, all `EmployeeAgent` instances register as “weak participants”. The `HumanResources` agent checks the skills of each employee, verifying if they possess the necessary capabilities to continue. If any employees lack required skills, the agent reports the number of underqualified employees back to the DirectionAgent. A LeaveDepartment event is then generated, instructing all agents to exit the Human Resources space. The DirectionAgent collects the number of employees needing training and sends this information to the PersonnelDepartment agent. The PersonnelDepartment agent formulates a training plan by issuing a Plan event, which outlines the specific training needs and objectives for the employees requiring upskilling.

Suppliers Phase The simulation then enters the suppliers phase, where the SupplierAgent adapts its behavior based on a NewStats event received from the DirectionAgent. This event carries information about the training plan and the specific requirements for the training session. The SupplierAgent modifies its characteristics (e.g., course speed, capacity, and content) to simulate different suppliers or training providers. Once it is ready, it generates a SupplierReady event. Employee agents join the supplier's space to participate in the training session. The SupplierAgent checks that all employees scheduled for training have arrived by comparing the number of participants in the space to the number expected. A LearnSkills event is triggered to allow employees to acquire the skills they lack. During this phase, employees update their skill sets based on the training plan. After training is complete, the SupplierAgent issues a LeaveDepartment event to have all the employees exit the supplier's space, signaling the end of the training phase for that iteration.

Superiors Phase The DirectionAgent communicates the number of trained employees to the Superiors agent, which oversees employee testing. The DirectionAgent sends the employees to the superiors' space for evaluation. The Superiors agent is responsible for verifying whether each employee has acquired the necessary skills from the training sessions. It checks the performance of each employee and assesses if they meet the required skill levels. After completing the evaluation, the Superiors agent generates a LeaveDepartment event, signaling that employees should exit the space and return to their default one. The DirectionAgent instructs all employees to reset to their initial state via a MakeClean event, preparing them for another iteration of the simulation if time allows. This ensures that the system can repeat the process if the simulation is set for multiple iterations.

Termination Phase Once the simulation's allocated time has expired, the DirectionAgent generates a Die event, instructing all agents to terminate. This event propagates to all agents (EmployeeAgent, SupplierAgent, DepartmentAgent, etc.), causing them to halt their execution and release any system resources they were utilizing. The simulation concludes with the graceful termination of all agents, and any simulation data, such as training outcomes or skill acquisition, is logged for analysis.

This process allows the simulation to mimic a real-world corporate training scenario, where employees are assessed for missing skills, trained by external suppliers, and evaluated by their superiors. The iterative nature of the simulation ensures that multiple cycles of training and assessment can be run, depending on the simulation parameters. Although the user cannot influence the simulation during its execution, changing the initial parameters will yield different results. By modifying the number of employees and the amount of skills they possess at the beginning, the duration of interactions is extended, and potentially a different number of cycles are executed within the defined timeframe. Changing the list of characteristics of course providers (course duration, cost, reliability, etc.) will result in agents learning at different rates and potentially requiring more courses to acquire the necessary skills. Observing the evolution of interactions between agents provides a perspective close to real training paths and is not limited to simply calculating a cost-benefit ratio. By observing the intermediate stages, it is possible to identify potential shortcomings of a training plan, reduce waiting times, or modify its characteristics, thus optimizing the physical and temporal resources required.

5. Implementation

In this section, we provide an brief overview of the main features of the agents implemented using SARL, detailing the code structure and their key functionalities[1]. Agents in the system communicate and coordinate with one another through various event-driven mechanisms, as illustrated in the following code excerpts.

DirectionAgent The `DirectionAgent` is responsible for initializing and managing the simulation, including creating various department and supplier agents. The agent also monitors the simulation duration, spawns departments, and triggers their interactions through event-based communication. The following code snippet shows how the `DirectionAgent` is initialized and starts the simulation.

```
agent DirectionAgent {

  var superiorSpace = defaultContext.getOrCreateSpaceWithID(typeof(OpenEventSpaceSpecification), ...)
  var superiorSpace = ...
  ...

  on Initialize {
    wake(new createSuperiors)
    wake(new createHumanResources)
    wake(new createPersonnelDepartment)
    wake(new createSupplier)
    wake(new createEmployees)
    in(simulationDuration) [
      var evt = new Die()
      // emit Die event
    ]
    ...
  }

  on createSuperiors { ... }
  on createHumanResources { ... }
  on createPersonnelDepartment { ... }
  on createSupplier { ... }
  on createEmployees { ... }

  on AgentSpawned [it.agentID != ID] {
    var n = this.count.incrementAndGet
    if (n === nEmployees + nDepartments + 1) {
      var evt = new NofEmployees(nEmployees)
      humanRSpace.emit(evt)
      ...
    }
  }

  ...

}
```

Listing 1: Excerpt of the `DirectionAgent`'s code.

The `DirectionAgent` manages key components such as `superiorSpace`, `humanRSpace`, and `personnelDSpace`, which are event spaces that handle inter-agent communication. Agents are spawned within specific spaces, allowing them to communicate and share events related to their roles within the simulation. The on construct is used to define the behavior of an agent in response to specific events. It allows an agent to listen for and handle events by specifying the event type and the actions to be taken when the event occurs.

EmployeeAgent The `EmployeeAgent` represents individual employees in the company, who can perform tasks, learn skills, and interact with departments. Each employee agent has the ability to join or leave departments and take part in training based on their lack of skills. Listing 2 reports an excerpt of the `EmployeeAgent`'s code.

[1] The full code of the system is available at https://di.unito.it/sarlcorporatetraining.

```
agent EmployeeAgent {

  uses Capacity1, Capacity2, ...

  on JoinDepartment { ... }
  on LeaveDepartment { ... }

  on LearnSkills {
    // Training logic for skill s1
    setSkillIfAbsent(new s1)
    ...
  }

  on TestSkills {
    try {
      // Test the newly acquired skill actions
    } catch (e : Exception) {
      // Training must be repeated
    }
    // If skills learnt, leave the space
  }

}
```

Listing 2: Excerpt of the `EmployeeAgent`'s code.

Employee agents can test and develop their skills, depending on whether they are found lacking, by interacting with departments such as `HumanResources`. The competences that the agents should acquire are managed by defining specific capacities, with agents getting the skills that implement them dynamically during the simulation.

DepartmentAgent The `DepartmentAgent` plays a key role in handling tasks for different departments within the organization. Each department is associated with a specific skill set required for its operation. The departments also monitor the number of employees and emit appropriate events when new employees join or leave. The following code shows the initialization of a `DepartmentAgent`.

```
on Initialize {
  switch (occurrence.parameters.get(1)){
    case "Superior": setSkillIfAbsent(new SuperiorSkill)
    case "Human Resources": setSkillIfAbsent(new HumanRSkill)
    case "Personnel Department": setSkillIfAbsent(new PersonnelDSkill)
    default: setSkillIfAbsent(new DefaultSkill)
  }
  ...
}
```

Listing 3: Initialization of `DepartmentAgent`.

Each department is assigned its unique skills, such as `SuperiorSkill` or `PersonnelDSkill`, depending on its functional role in the company. These skills define the specific behavior of the department and how it interacts with employees and other departments.

SupplierAgent The `SupplierAgent` is responsible for managing interactions with external suppliers, including the emission of events such as `SupplierReady`. This agent monitors the number of participants (e.g., employees) and adjusts its behavior based on received event data in order to simulate different training patterns. An excerpt of `SupplierAgent`'s code is reported below.

```
agent SupplierAgent {

  var nStudents : Number
  var speed : long
  var probability : int

  on NewStats {
    var plan = occurrence.plan
    nStudents = plan.NDoc
    speed = plan.speed
    probability = plan.probability
    var evt = new SupplierReady()
    evt.source = comspace.getAddress(getID())
    comspace.emit(evt)
    wake(evt)
  }

  on ParticipantJoined [(!isFromMe) && (occurrence.spaceID == comspace.spaceID)] {
    synchronized (sem){
      if (nStudents != 0 && comspace.numberOfWeakParticipants == nStudents && !executed) {
        var learn = new LearnSkills(probability)
        learn.source = comspace.getAddress(getID())
        comspace.emit(learn)

        executed = true
        in(speed) [
          var leave = new LeaveDepartment(comspace, true)
          leave.source = comspace.getAddress(getID())
          comspace.emit(leave)
          executed = false
        ]
      }
    }
  }

}
```

Listing 4: Excerpt of the `SupplierAgent`'s code.

The `SupplierAgent` handles the coordination of learning and participation in a supplier context, where it monitors student readiness and triggers the appropriate actions.

Event-Driven Architecture The system heavily relies on an event-driven architecture where different agents communicate by emitting and handling events. Events such as `Die`, `JoinDepartment`, and `LearnSkills` enable coordination and interaction between agents. The following is a sample event declaration.

```
event LearnSkills {
    val learningProbability : int

    new(learningProbability : int) {
        this.learningProbability = learningProbability
    }
}
```

Listing 5: Event declaration example.

These events are used across the system to trigger actions, update states, and enable communication among agents.

Skill Management Agents in the system can possess or acquire skills through predefined capacities. The following excerpt illustrates, e.g., how the `SuperiorSkill` manages employees' skills.

```
skill SuperiorSkill implements StandardCapacity {

    var nEmployees = 0
    var executed = false

    synchronized def checkGuests(comspace : OpenEventSpace) {
        if (nEmployees != 0 && comspace.numberOfWeakParticipants == nEmployees && !executed) {
            executed = true
            var evt = new TestSkills(comspace)
            evt.source = comspace.getAddress(getID())
            comspace.emit(evt)
            in(1000) [
                if (comspace.numberOfWeakParticipants != 0) {
                    var problem = new NofEmployees(comspace.numberOfWeakParticipants)
                    problem.source = comspace.getAddress(getID())
                    comspace.emit(problem)
                }
                var leave = new LeaveDepartment(comspace, true)
                leave.source = comspace.getAddress(getID())
                comspace.emit(leave)
                executed = false
            ]
        }
    }

}
```

Listing 6: `SuperiorSkill` example.

Each department has its own unique set of skills, enabling the simulation to emulate complex organizational structures and workflows.

6. Discussion and conclusion

The aim of this study was to demonstrate the potential of multi-agent systems to enhance corporate AI training, particularly in remote work settings. The research addressed critical issues in corporate AI training, especially in remote and hybrid work contexts, where traditional in-person training methods fall short. A MAS-based solution provides a flexible, scalable, and adaptive system that simulates training processes to foster AI literacy among employees, regardless of their geographical location. The proposed system, built using the SARL agent programming framework, allows organizations to simulate complex training workflows, accommodating varying skill levels and needs of employees. This adaptive and scalable solution addresses key challenges in remote working environments, such as delivering personalized learning experiences, and providing real-time feedback.

The system’s architecture, with autonomous agents representing employees, departments, and external training providers, enables dynamic simulations, which aim at mirroring real-world training scenarios and organizational processes. It allows companies to simulate different training plans without additional time or financial investments, providing estimates of potential gains from improved employee skills. In this setting, the collection of real data about providers could involve only a small sample of employees concretely taking the courses and comparing their pre- and post-training knowledge. By considering their feedback, companies could estimate provider characteristics, input them as simulation parameters, and test the effectiveness of training programs on a larger number of employees represented as SARL agents. Preliminary results from the simulation highlight the effectiveness of a MAS-based approach in reducing training time and costs while ensuring that employees acquire essential AI skills.

One of the most significant contributions of this work is the ability of the system to model different training scenarios and predict outcomes based on data-driven insights. By offering companies the ability to test various training strategies without investing additional resources, the system empowers decision-makers to optimize training paths and improve overall employee performance.

Several future developments may enhance the presented system’s capabilities. For instance, future work could include incorporating machine learning algorithms to analyze historical training data and improve the simulation’s accuracy, as discussed in [10]. By learning from previous results, the system could predict employee training outcomes. This could lead to more accurate forecasts of training success and help organizations fine-tune their AI training programs. Furthermore, the creation of a user-friendly interface using tools like JavaFX[2] would make the system more accessible to non-technical users. This interface could allow administrators to input parameters such as number of employees, training duration, course characteristics, and skill gaps without modifying the underlying code. A more intuitive UI would also make it easier to interpret the simulation results. Future versions of the system could connect to live databases, pulling real-time data about employee performance and training needs. This would enable continuous updates to the simulation model and offer more accurate and relevant results, ensuring that the training plans reflect current organizational requirements. Additional functionalities could be implemented in the agents to simulate more complex training workflows and interactions. For example, agents could be designed to represent different managerial levels, allowing

[2] https://openjfx.io/.

for simulations of leadership training or cross-departmental collaboration. Finally, the development of a companion tool for employees could provide them with personalized guidance through the training workflow. By capturing individual performance data and feeding it into the simulation, the system could model employee progress more accurately and provide real-time feedback on their learning process.

In conclusion, the presented system represents a first step to promote the role of MAS in supporting remote and hybrid workforce training, making them a powerful tool for modern organizations aiming to foster AI literacy and awareness.

Acknowledgments

This publication is part of the project NODES which has received funding from the MUR – M4C2 1.5 of PNRR funded by the European Union - NextGenerationEU (Grant agreement no. ECS00000036).

References

[1] A. Rao, B. Greenstein, AI Predictions 2022: Six AI Trends Shaping Business, Technical Report, PwC, 2022.

[2] M. Chui, L. Yee, B. Hall, A. Singla, A. Sukharevsky, The State of AI in 2023: Accelerating Breakthroughs and Adoption, Technical Report, McKinsey & Company, 2023.

[3] S. Ransbotham, D. Kiron, P. Gerbert, M. Reeves, Reshaping business with artificial intelligence: Closing the gap between ambition and action, MIT Sloan Management Review 59 (2017).

[4] P. Castelli, Pianificazione della formazione: processi, attori e strumenti, Technical Report, EBC Consulting, 2012. URL: https://www.ebcconsulting.com/images/filePAT_pdf/h1_hrms_e_pianificazione_formazione.pdf.

[5] S. Suravi, Training and development in the hybrid workplace, The Learning Organization 31 (2024) 48–67.

[6] O. Olawale, F. A. Ajayi, C. A. Udeh, O. A. Odejide, Remote work policies for IT professionals: review of current practices and future trends, International Journal of Management & Entrepreneurship Research 6 (2024) 1236–1258.

[7] Z. Pokojski, A. Kister, M. Lipowski, Remote work efficiency from the employers' perspective—What's next?, Sustainability 14 (2022) 4220.

[8] M. Lovcheva, V. Konovalova, M. Simonova, Development of corporate digital training, in: Digital Age: Chances, Challenges and Future 7, Springer, 2020, pp. 473–479.

[9] M. Wooldridge, An introduction to multiagent systems, John Wiley & Sons, 2009.

[10] X. Chen, H. Chen, J. Xu, An innovative approach to corporate hr training based on deep learning, Applied Mathematics and Nonlinear Sciences 9 (2024).

[11] S. Rodriguez, N. Gaud, S. Galland, SARL: a general-purpose agent-oriented programming language, in: 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), volume 3, IEEE, 2014, pp. 103–110.