LAMAS: An Organisation-centred Architecture for Agentic AI

Babette Dellen[1], Michael Köhler-Bußmeier[2], Wied Pakusa[1] and Jan Sudeikat[2]

[1]Hochschule Koblenz, Konrad-Zuse-Str. 1, D-56075 Koblenz, Germany
[2]Hamburg University of Applied Sciences, Berliner Tor 7, D-20099 Hamburg, Germany

Abstract

Current frameworks such as *LangGraph*, *CrewAI*, and *OpenAI Swarms* enable the modelling of static LLM-based multi-agent systems (MAS). However, these systems remain unreliable in dynamic environments due to the lack of clear organisational structures, transparent coordination, and adaptability.

The presented project *Learn2Organise* addresses these limitations by developing an open-source architecture, called LAMAS, that supports the creation and execution of learning, self-adaptive, and organisation-centred LLM-MAS. A core idea of LAMAS is to complement LLM agents within a symbolically defined organisation to improve explainability and reliability.

LAMAS does not only support the the specification of agent organisations (with roles and communication flows) and the deployment of LLM-MAS into real-world application processes; it also has a execution engine which enables self-adaptation of organisational structures at run-time through reinforcement learning, including dynamic role assignment and communication path adjustments. Additionally, it also uses process mining techniques to extract interaction and decision structures from log data, automatically generating organisational models and monitoring them at run-time.

Keywords

Organisation, multi-agent systems, self-adaptation, Q-learning, process mining

1. Learn2Organise: Agentic AI and Organisation-Centred MAS

Large Language Models (LLMs) based on the attention mechanism [1] and derived (LLM)-agents are fundamentally transforming our working environment. LLMs are deep learning models based on transformer architectures [1], trained on large datasets to process, understand, and generate natural language. Well-known LLMs include ChatGPT, Gemini, and Claude, which already handle and solve complex tasks today.

This opens up new application areas for AI systems in the fields of digitalisation and automation. However, leveraging the potential of LLM-agents in practice requires effective integration into (existing) workflows and organisational structures. Real-world problems almost always demand *organised collaboration* among multiple *specialised experts*, who may also need to adhere to highly *regulated processes*.

Initially, LLMs have no inherent capability to autonomously interact with dynamic environments or other agents, nor to plan actions [2], as they were simply not trained for this. In fact, several studies show that autonomous organisation of (current) LLM-agents is highly likely to fail, e.g., [3, 4]. LLMs are not designed as goal-directed actors but as probabilistic token generators. Accordingly, modern LLM-based multi-agent systems (LLM-MAS) *without* well-thought-out organisational structures usually perform no better than individual LLM-agents in leading benchmarks [5].

We argued in [6] that Agentic AI, which could be considered as a discipline of 'sub-symbolic' AI, is faced with similar problems that the MAS community, which is more related to 'symbolic' AI, long

PNSE'26, International Workshop on Petri Nets and Software Engineering, 2026

† These authors contributed equally.

dellen@hs-koblenz.de (B. Dellen); michael.koehler-bussmeier@haw-hamburg.de (M. Köhler-Bußmeier); pakusa@hs-koblenz.de (W. Pakusa); jan.sudeikat@haw-hamburg.de (J. Sudeikat)
https://www.haw-hamburg.de/michael-koehler-bussmeier (M. Köhler-Bußmeier); https://www.haw-hamburg.de/jan-sudeikat (J. Sudeikat)
0000-0002-3074-4145 (M. Köhler-Bußmeier); 0009-0004-6302-4445 (W. Pakusa); 0009-0007-7871-9006 (J. Sudeikat)

before. Classical MAS and organisational structures were intensively studied between 1990 and 2010, cf. [7, 8]. Therefore, we proposed to adapt 'old' recipes from the MAS community to the needs of modern LLM-agents. Specifically, our approach relies on the *Organisation-Centred Multi-Agent Systems (OCMAS)* framework [9], where not only agents but also the overall system structure (organisational chart, positions, roles, goals, tasks, etc.) is considered. In an OCMAS, the macro-structure (organisation) integrates with the micro-elements (agents). The major advantage is that the organisation defines a corridor of permissible actions for agents, preventing forbidden actions and improving collaboration (e.g. using *guardrails*).

Since organisations should not be static structures, we advocated so called "learning organisations". More concretely, we advocate *adaptive MAS structures* that dynamically adjust their organisation at run-time.

Such self-adaptation has two sides: A 'sensing' side, which evaluates the running system and a 'acting side, which plans the adaptation. Our *Learn2Organise* approach [6] combines ideas from classical MAS research with *process mining* [10], which is very popular in the Petri net community, for the sensing and *(deep) reinforcement learning* [11] for the planning. We identified the following core functionalities:

- **DESIGN:** A modelling tool for MAS organisations, including analysis tools (cf. [12, 13]).
- **EXECUTE:** A virtual environment for agents and OCMAS, including deployment and feedback loops for learning and adaptation (cf. [14]).
- **MINE:** Automated derivation of OCMAS designs from log data and alignment of current vs. target states using Process Mining techniques [10] and model simulations [15].
- **ADAPT:** Self-(re)organisation driven by feedback loops and Reinforcement Learning variants [11].

In this paper we present the architecture of our framework, called Lamas, that enables the core functionality. Lamas enables the *structured organisation* of LLM-agents and leverages *data-driven analysis techniques* from *process mining (PM)* to analytically account for environmental dynamics during both design and execution of MAS.

The paper has the following structure. In Section 2 we recap our argument from [6] why agentic AI has an urgent need for the concept of organisations to ensure a *reliable* kind of flexibility. We will describe the core aspects of our *Learn2Organise* approach (namely: design, execute, mine, and adapt), which combine agentic AI (based on LLMs) with adaptive MAS based on organisation models. In Section 3 we present the architecture of Lamas, especially the Lamas run-time engine which enables all system activities within a feedback-loop, including learning of agents and self-adaptation of the organisation. The work ends with a conclusion.

2. Agentic AI and the Need for Adaptive Organisations

Currently, a wide range of frameworks for Multi-Agent Systems (MAS) based on Large Language Models (LLMs) is emerging (e.g., *LangGraph*, *CrewAI*, *OpenAI Swarms*, or n8n) [16, 17, 18]. The application domains are highly diverse, ranging from software development [19], through robotics [20], to the simulation of complex social systems [21].

In practical deployments (as confirmed by our industry partners), it becomes evident that *successful cooperation* among LLM agents requires a *well-founded and carefully designed* organisational structure—analogous to real-world organisations. Such a (symbolic) structure also helps address questions of behavioural traceability, a practical challenge extensively studied under terms such as *Explainable AI* and *Responsible AI* (cf. [22] and [23]).

Agent systems (in the sense of *Agentic AI* based on *LLM/Generative AI* [1]) are, of course, not a new phenomenon—the research on Multi-Agent Systems (MAS) dates back to the 1990s. Among other aspects, *autonomy* was intensively investigated during the peak of MAS research (roughly between 1990 and 2010, see also [7, 8]). The obvious question we raised in [6] is therefore: *Which previously developed approaches to address these challenges can be reused for current* Agentic AI*, and which aspects of LLM agents require new measures?*

A fundamental problem in MAS arises from the fact that each agent pursues its own goals; the *prisoner's dilemma* from game theory [24] illustrates that locally rational actions generally do not lead to globally desirable outcomes. Coherent behaviour must therefore be laboriously negotiated, which is only feasible for a small number of agents (scalability issue). Moreover, coordination raises questions of *controllability* and *predictability*, i.e., developers typically cannot foresee which common goal the agents will ultimately agree upon. For larger numbers of agents, structures are needed that restrict the agents' action space from the outset, and such structures are found in *MAS organisations* (with positions, rights, a network of roles with aligned goals, and roughly predefined action plans, see [9, 23]). MAS organisations provide the required *symbolic layer* that constrains the behaviour of (mostly *sub-symbolic*) agents to achieve goal-oriented actions.

Nevertheless, tight monitoring remains essential for LLM-based MAS. Research in this area is still in its infancy, especially regarding crucial questions of self-reflection (i.e., using own observations) and adaptation, see [25]. The project *Learn2Organise* therefore builds on proven and well-studied methods from the field of *Process Mining* [10]. Process mining deals with data-driven analysis of process flows based on event logs generated by information systems. These logs consist of events that can be temporally ordered, thus representing sequential or parallel process steps. The goal of process mining is to algorithmically exploit this information to monitor, optimise, and detect deviations from desired process specifications. In the context of LLM-MAS, we aim to use process mining to (1) automatically learn models of OCMAS from log data [26, 27] and (2) monitor and adapt current models (e.g., to improve performance or meet organisational specifications), see [28]. To achieve this, we follow established feedback loops such as the *MAPE-K loop* (*Monitor-Analyse-Plan-Execute with Knowledge* [29, 30]). A MAPE-based system continuously monitors key metrics (also known as *key performance indicators (KPI)*, analyses whether adjustments are necessary, plans an appropriate response, and then executes the plan. For this purpose, recent techniques of *object-centric process mining* also appear promising [31].

The *self-adaptation* of agents during run-time to new tasks and environments is studied under the term *self-organising systems* [32]. Self-reorganisation of MAS can be enabled through *multi-agent reinforcement learning* [33]. Each agent is driven by rewards and interacts with other agents. Both the information exchange among agents and the complexity of LLMs result in high-dimensional state spaces and input data that agents must translate into actions. Deep Q-learning uses deep neural networks to learn successful strategies from experience [34, 35]. Ideally, the model should generalise to previously unseen situations, which are highly likely due to the complex dynamics of MAS. This imposes significant requirements on neural networks. It is therefore an open research question how deep Q-learning can be effectively applied to systems of LLM agents.

3. Enabling Organisations of LLM-Agents with Lamas

The *Learn2Organise* approach project combines principles of self-organisation in MAS, guided by the strategy *Design-Execute-Mine-Adapt*, with data-driven analyses of organisational structures. This enables the integration of LLM agents into modern business processes.

The interplay of the concepts is sketched in Fig. 1. We start with an *organisational model* (in the block named *DESIGN*). This model is deployed in the running *OCMAS* and an *OCMAS clone*, which is an abstract, simplified version of the OCMAS. The *OCMAS* is executed on an engine, the *OCMAS-VM* (in the block *EXECUTE*). The *observations* stimulate the *learning* and *organisational self-adaptation* (in the block *ADAPT*). From the event *logs* we mine the *'real' (organisational) design*. Additionally, we use the *OCMAS clone* to *analyse* by simulation how possible adaptation steps would effect given key performance indicators of the OCMAS. Both processes – mining and simulation – trigger a *re-design* of the organisation model (in the block *DESIGN*), which closes the feedback loop.

3.1. The OCMAS Architecture Lamas

The process *Design-Execute-Mine-Analyse* relies on the Lamas Architecture which is shown in Fig. 2. The architecture is organised as stacked layers; upper layers are built upon the facilities and abstractions

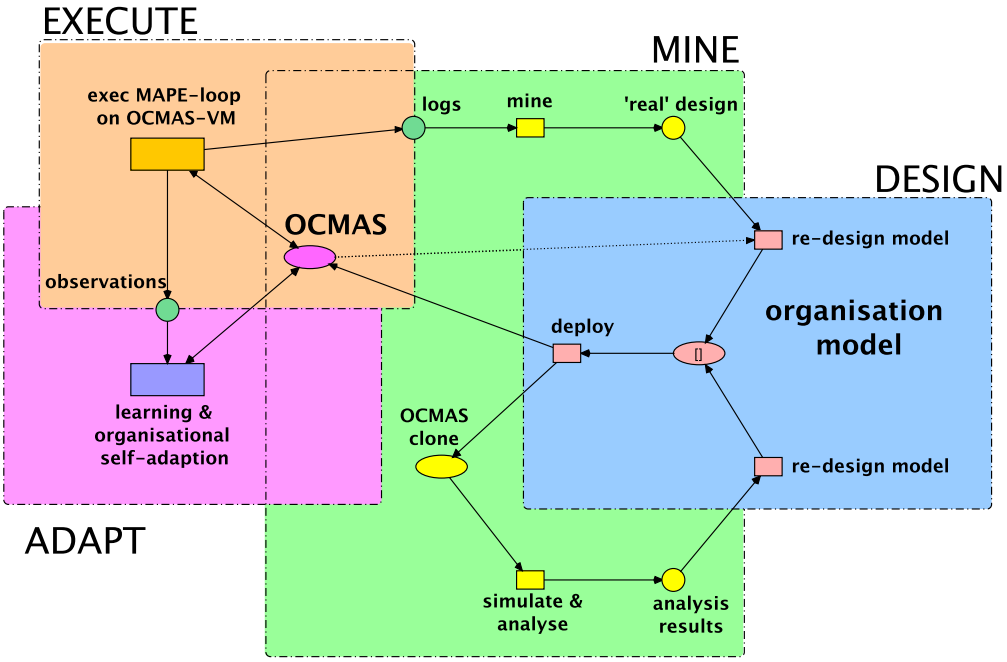

Figure 1: The Interplay of the Project Core Aspects: Design-Execute-Mine-Analyse (adapted from [6])

of the lower ones. The base layer is the *Distributed Agent Platform*, which provides typical services for multi-agent systems, like agent creation, messaging, and deployment on machines.

On top of the base level we add a run-time execution model based on the MAPE-K pattern. This layer structures how agents work together. Here, the Knowledge K incorporates the organisational model which structures every step of the loop. This layer provides the shared semantics, reusable assets, tuneable parameters, and organisational structure that ground all reasoning and adaptation.

The MAPE Loop itself incorporates the *Org-Model*. Agents perceive the environment and their peers via *Sensors* and influence them via *Actuators*. Sensors are used in the monitoring phase and actuators during the execution phase of the MAPE loop. It also contains the specification of the *KPIs/Transformation-Operators*. These elements are used in self-adaptation processes of the organisation. The key performance indicators (KPI) acts as a kind of sensor as they monitor some relevant aggregated state of the organisation and the transformation-operators act as special actuators that operate on the internal organisational run-time model. Both, sensors and KPI are used by the *trigger rules*, usually in combination with run-time *knowledge*, to start multi-agent interaction specified by *agent interaction protocols (AIP)*. AIPs specify the message patterns and roles. We assign orders to AIPs. A first-order AIP is a normal protocol with effects on the agents' state or the environment. A second-order AIP is a protocol that is used for self-adaptation. It is carried out by the agents in a coordinated interaction process the same way as for first-order AIP, but the protocol modifies the organisational model. The execution of interactions is called *Teamwork*, which describes the inter-agent processes. Teamwork governs group negotiation of constraint, and resolution of conflicts. During the execution of teamwork agents update their behaviour due to learning etc. This update process is encapsulated in the so called *Agent-Life-Cycle*.

The MAPE-K Loop contains a special sub-system, the *Organisation-Simulator@Run-Time* [36], which uses the organisation model to generate a digital twin of the OCMAS. This simulator is used during the planning phase to predict the impact of transforming the organisation in case of second-order self-adaptation of the MAS [37, 38, 39, 40]. The simulator relies on a model of those system parts, which include the environment, but also the agents as well. The twin model has to be an abstraction since the planning step of MAPE has to analyse itself which clearly raises performance and decidability issues. Our organisational model provides a 'natural' candidate for abstraction: the *position/member interface*.

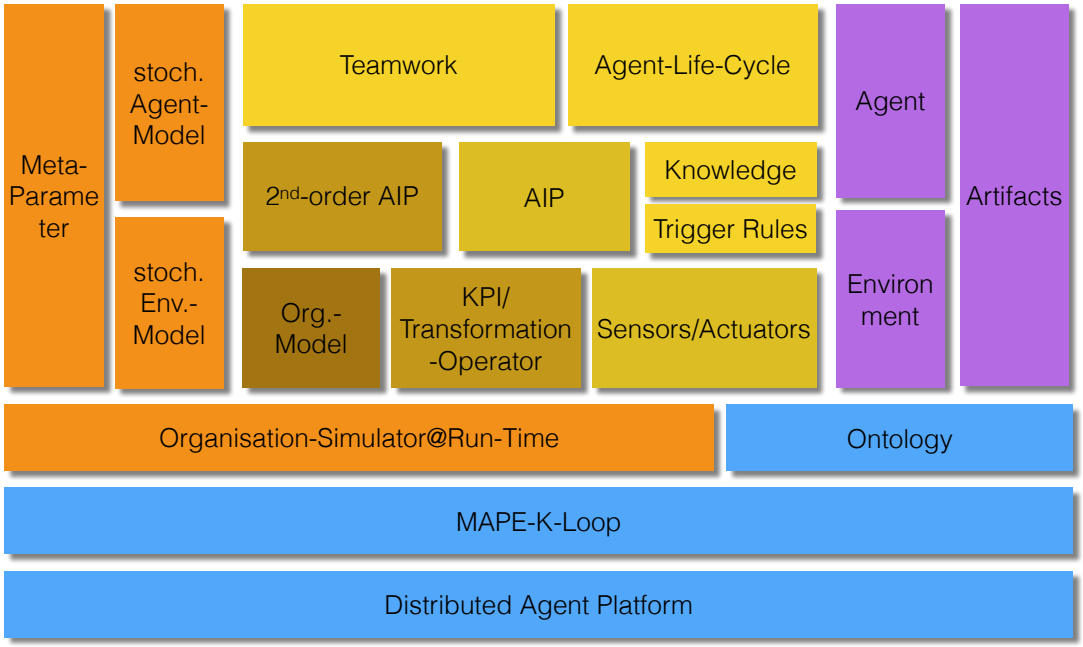

Figure 2: The LAMAS Architecture

As explained in the following section in more detail our execution engine represents the organisational model using a network of so called *position* agents. These agents supervise the organisational constraints at run-time, but they do not implement the organisational roles; this is done by the *member* agents. So we have a *Janus-faced* couple of position and member agents . Usually, this allows to decouple the organisational specification and the implementation, e.g., in an organisation for a *market place* we have position agents for *providers* and *bidders* and define their interaction within a *auction protocol*. The concrete implementation of bidder is part of the member agent and is hidden from the organisation. So, from the perspective of the organisation member agents are black boxes. So it is obvious to abstract at the interface of the member agents. For simplicity we use a *Stochastic Environment Model* and *Stochastic Agent Models* since statistical data about agents is collected during the Monitoring-Phase of in MAPE anyway. We also abstract from the position-member interaction which integrates the organisational constraints with the member's desires. This interaction is integrated into the stochastic model of the member as meta-parameters of the probability distributions. For example we have the so called *organisational impact* c_{org} which describes how ‚eagerly' members follow the positional constraints. The stochastic model is the counterpart of the real system, i.e. the *Environment* and the *Agents*) together with the *Artefacts*.

3.2. The LAMAS Run-Time Engine

We now take a deeper look at the run-time loop itself. The LAMAS Run-Time Engine enabling the organisation model is shown in Fig. 3. In the following we are going to explain the structure of the Petri net shown, but we will not explain all details here, e.g. we will not dive into the inscriptions of arcs of guards. The engine is based on the SONAR model [41, 42, 36]. In SONAR we have two kinds of agents: The *organisational position agents (OPA)* and the *organisational member agents (OMA)*. Each OPA defines a position in the organisational network and is aware of the rights and obligations assign to it. Each position is filled (or: implemented) by one OMA which must be capable to to implement the position. The set of OPAs is defined by the organisational model and remains fixed unless the model is changed. The set of OMAs is more flexible as each OMA is assigned to its OPA and OMAs may resign ('hire and fire'). As a rough analogy, an OPA is more like a type or interface and an OMA is like an implementation.

The run-time engine is formalised as a Petri net (shown in Fig. 3) and is structured into three conceptual regions: (i) the agent life-cycle and OPA/OMA agent management (upper left), (ii) the organisational repository (lower left), and (iii) the MAPE loop (right side). The behaviour may be described as three interacting subsystems. The loop integrates organisational agents (position agents, called OPAs, and member agents, called OMAs), an organisational repository, and a full monitor–analyse–plan–execute cycle. The organisational model govern how teams form and interact and how organisational change is carried out as a kind of self-adaptation.

OPA/OMA Management The management of agents at run-time is given in the upper-left part of Fig. 3. The organisational consists of a network of positions. Each position specifies a set of permissions, obligations, protocols and roles. Each position is expressed by a special agent, the *organisational position agents* (OPA). Each position agent is connected to an *organisational member agent* (OMA). Members are a kind of implementation, while positions are a kind of specification. In general a member could be a human (connected via a user interface), a classical software systems (executing if-then rules), or, as in considered here, LLM-based agents.

The connection of members and positions is a dynamic one. Transitions such as *assign* create a connection of an *sub-organisations or agents* with *unassigned OPA*. This assignment make the external agent a member of the organisation, which is stored at *OMAs*, and the OPA is now an *assigned OPA*. OMAs and OPAs have *internal communication* to guarantee that the OMA satisfies the role specification. We omit details, which are modelled as special internal protocols.

Organisational Repository (Adaptation of the Organisational Model) The organisational model, containing observables, trigger rules, team operators, interaction protocols, etc., is stored in a repository which is given in the lower-left part of Fig. 3.

Each of these model components can be modified through transitions like *modify observables*. These transitions carry labels like transform("first_order") that are synchronisation channels, i.e., we have transitions from the MAPE loop part on the right hand side that are activated during the execution of plans and these transitions synchronise with the transformation of the organisational model. These transitions implement first- and second-order adaptations. Here, first-order adaptations modify the environment via actuators, while second-order adaptations change the organisational run-time model itself: its own rules, performance indicators, negotiation schemes, and coordination structures. The repository therefore supplies the MAPE loop with dynamic semantics, which the loop may also reconfigure as part of execution.

The MAPE-K Loop The right-hand side of Fig. 3 models the operational MAPE-cycle. All events like *monitor*, *analyse & trigger*, etc. are carried out by OPAs in a coordinated way. During the execution each OPA internally coordinates with its OMA.

The *monitor* transition collects world-state information, producing *observations*. The monitoring is carried out by the OPAs and relies on the *observables* given by the organisation model, like the sensors of the organisation. Using the repository's trigger rules (TR), the *analyse & trigger* transition determines whether adaptation is required, producing an initial task and parameters such as team constraints and selected AIPs.

The planning phase consists of two major transitions: *form team*, where team-formation operators assemble or modify the current teams, and *negotiate*, in which team members negotiate the interaction protocols and the run-time constraints.

The *execute* transition applies first-order and second-order transformations depending on the concrete task. Some observation trigger 'normal' plans, so called first-order actions. The execution of those use actuators to transform the *world state* of the environment. The basic interaction is specified by so-called *agent interaction protocols* (AIP).

Other observations demand for a self-adaptation of the organisation itself. Self-adaptation is carried out the same way as 'normal' activities. It is described by second-order AIP and the effect of *second-*

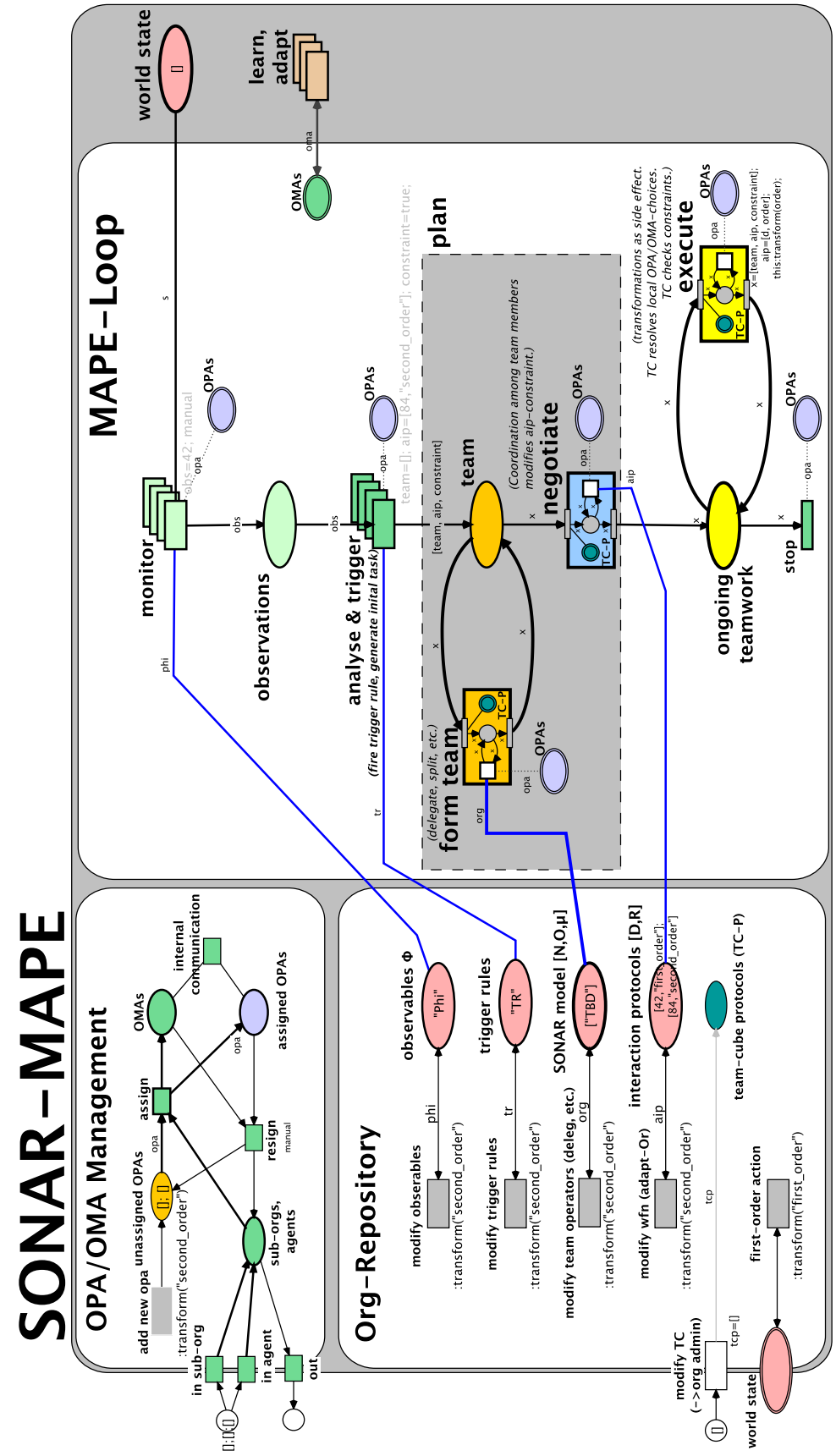

Figure 3: The LAMAS Run-Time Engine: Agent Management, Repository of the Organisation Model, and the MAPE Feedback Loop (Monitor, Analyse, Plan, and Execute)

order actuators is an transformation of the organisational model using the synchronous channels *:transform("second_order")*. This may include updating trigger rules, the organisation model, or the interaction protocols.

The interplay between first-order (operational) and second-order (meta-level) transitions enables self-adaptation of both behaviour and organisational structure at run-time.

4. Conclusion

LLM agents have great potential for boosting productivity; however, they also pose a significant challenge for predictability (in terms of *compliance*). While reliability and explainability are important, they should not dominate the narrative, as our main argument is that poorly organised MAS simply fail to perform effectively.

Here we presented the *Lamas Architecture* which integrates and enables the basic phases of our approach: *Design*, *Execute*, *Mine*, and *Adapt*.

To make MAS predictable, Lamas uses organisations as a design element. Structured organisational models provide coordination and control mechanisms that ad-hoc agent configurations lack. Organisations should not be completely rigid structures, as this can be paralysing. To ensure flexibility, we support *learning organisations*, i.e., adaptive organisational models based on the OCMAS paradigm that can evolve at run-time. When OCMAS adapt, the actual system state (*'as it is'*) may diverge from the intended design (*'as it should be'*). Therefore, Lamas continuously monitors the execution and applies process mining techniques to realign the system.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, Curran Associates Inc., Red Hook, NY, USA, 2017, p. 6000–6010.

[2] H. Wei, Z. Zhang, S. He, T. Xia, S. Pan, F. Liu, Plangenllms: A modern survey of llm planning capabilities, 2025. URL: https://arxiv.org/abs/2502.11221. arXiv:2502.11221.

[3] G. Piatti, Z. Jin, M. Kleiman-Weiner, B. Schölkopf, M. Sachan, R. Mihalcea, Cooperate or collapse: Emergence of sustainable cooperation in a society of LLM agents, in: NeurIPS, 2024.

[4] S. Abdelnabi, A. Gomaa, S. Sivaprasad, L. Schönherr, M. Fritz, Cooperation, competition, and maliciousness: Llm-stakeholders interactive negotiation, in: NeurIPS, 2024.

[5] M. Z. Pan, M. Cemri, L. A. Agrawal, S. Yang, B. Chopra, R. Tiwari, K. Keutzer, A. Parameswaran, K. Ramchandran, D. Klein, J. E. Gonzalez, M. Zaharia, I. Stoica, Why do multiagent systems fail?, in: ICLR 2025 Workshop on Building Trust in Language Models and Applications, 2025. URL: https://arxiv.org/abs/2503.13657.

[6] B. Dellen, M. Köhler-Bußmeier, W. Pakusa, J. Sudeikat, Learn2organise: Organisational design for reliable adaptive systems of llm-agents, in: Workshop on Robust Applied Intelligent Assistive Systems for Society & Enterprise (RAISE), Springer Nature Switzerland, Cham, 2026.

[7] G. Weiß (Ed.), Multiagent systems: A modern approach to Distributed Artificial Intelligence, MIT Press, 1999.

[8] G. Weiß (Ed.), Multiagent systems (2nd edition), MIT Press, 2013.

[9] V. Dignum, J. Padget, Multiagent organizations, in: G. Weiss (Ed.), Multiagent Systems, 2nd ed., Intelligent Robotics; Autonomous Agents Series, MIT Press, 2013, pp. 51–98.

[10] W. M. P. van der Aalst, Process Mining: Data Science in Action, 2 ed., Springer, Heidelberg, 2016. doi:10.1007/978-3-662-49851-4.

[11] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, The MIT Press, 2018.
[12] J. Sudeikat, M. Köhler-Bußmeier, On combining domain modeling and organizational modeling for developing adaptive cyber-physical systems, in: Proceedings of the 14th International Conference on Agents and Artificial Intelligence - Volume 1: ICAART,, INSTICC, SciTePress, 2022, pp. 330–336. doi:10.5220/0010881200003116.
[13] M. Köhler-Bußmeier, J. Sudeikat, Defining adaption measures for organizational multi-agent systems, International Journal of Parallel, Emergent and Distributed Systems (2023). doi:10.1080/17445760.2023.2243005.
[14] J. Sudeikat, M. Köhler-Bußmeier, Assets, artifacts and multi-agents: On enacting adaptive organizations in cyber-physical systems, in: Informatik 2024, Lecture Notes in Informatics, 2024, pp. 1887–1892. URL: https://doi.org/10.18420/inf2024_163, 9th IACS Workshop 2024.
[15] N. Bencomo, S. Götz, H. Song, Models@run.time: a guided tour of the state of the art and research challenges, Software & Systems Modeling 18 (2019) 3049–3082.
[16] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, J. Wen, A survey on large language model based autonomous agents, Frontiers of Computer Science 18 (2024). doi:10.1007/s11704-024-40231-1.
[17] X. Li, S. Wang, S. Zeng, Y. Wu, Y. Yang, A survey on llm-based multi-agent systems: workflow, infrastructure, and challenges, Vicinagearth 1 (2024). doi:10.1007/s44336-024-00009-2.
[18] T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, X. Zhang, Large language model based multi-agents: A survey of progress and challenges, in: K. Larson (Ed.), Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24, International Joint Conferences on Artificial Intelligence Organization, 2024, pp. 8048–8057. URL: https://doi.org/10.24963/ijcai.2024/890. doi:10.24963/ijcai.2024/890, survey Track.
[19] S. Hong, M. Zhuge, J. Chen, X. Zheng, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou, C. Ran, L. Xiao, C. Wu, J. Schmidhuber, Metagpt: Meta programming for A multi-agent collaborative framework, in: ICLR, OpenReview.net, 2024.
[20] Y. Chen, J. Arkin, Y. Zhang, N. Roy, C. Fan, Scalable multi-robot collaboration with large language models: Centralized or decentralized systems?, in: ICRA, IEEE, 2024, pp. 4311–4317.
[21] J. S. Park, J. C. O'Brien, C. J. Cai, M. R. Morris, P. Liang, M. S. Bernstein, Generative agents: Interactive simulacra of human behavior, in: UIST, ACM, 2023, pp. 2:1–2:22.
[22] A. Ricci, S. Mariani, F. Zambonelli, S. Burattini, C. Castelfranchi, The cognitive hourglass: Agent abstractions in the large models era, in: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS '24, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2024, p. 2706–2711.
[23] U. M. Borghoff, P. Bottoni, R. Pareschi, An organizational theory for multi-agent interactions integrating human agents, llms, and specialized AI, Discov. Comput. 28 (2025) 138.
[24] Y. Shoham, K. Leyton-Brown, Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations, Cambridge University Press, New York, 2008.
[25] M. Renze, E. Guven, Self-reflection in LLM agents: Effects on problem-solving performance, CoRR abs/2405.06682 (2024).
[26] A. Tour, A. Polyvyanyy, A. A. Kalenkova, Agent system mining: Vision, benefits, and challenges, IEEE Access 9 (2021) 99480–99494.
[27] R. H. Bemthuis, S. Lazarova-Molnar, Discovering agent models using process mining: Initial approach and a case study, in: ISPA/BDCloud/SocialCom/SustainCom, IEEE, 2022, pp. 163–172.
[28] C. W. Gunther, S. Rinderle-Ma, M. Reichert, W. M. Van Der Aalst, J. Recker, Using process mining to learn from process changes in evolutionary systems, International Journal of Business Process Integration and Management 3 (2008) 61–78.
[29] D. Weyns, Engineering self-adaptive software systems – an organized tour, in: 2018 IEEE 3rd International Workshops on Foundations and Applications of Self* Systems (FAS*W), 2018, pp. 1–2. doi:10.1109/FAS-W.2018.00012.
[30] D. Weyns, An Introduction to Self-Adaptive Systems: A Contemporary Software Engineering Perspective, John Wiley & Sons Ltd, 2020.

[31] W. M. P. van der Aalst, Object-centric process mining: An introduction, in: ICTAC Summmer School, volume 13490 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 73–105.
[32] J. O. Kephart, D. M. Chess, The vision of autonomic computing, IEEE Computer 36 (2003) 41–50.
[33] Z. Ning, L. Xie, A survey on multi-agent reinforcement learning and its application, Journal of Automation and Intelligence 3 (2024) 73–91. URL: https://www.sciencedirect.com/science/article/pii/S2949855424000042. doi:https://doi.org/10.1016/j.jai.2024.02.003.
[34] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (2015) 529–533. URL: https://api.semanticscholar.org/CorpusID:205242740.
[35] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of go with deep neural networks and tree search, Nature 529 (2016) 484–489. doi:10.1038/nature16961.
[36] M. Köhler-Bußmeier, J. Sudeikat, H. Rölke, L. Capra, Petri nets as run-time models for self-adaptive cyber-physical systems, in: Petri Nets and Software Engineering 2024, PNSE'24, volume 3730, CEUR, 2024. URL: https://ceur-ws.org/Vol-3730/paper10.pdf.
[37] L. Capra, M. Köhler-Bußmeier, Modelling adaptive systems with nets-within-nets in maude, in: Proceedings of the 18th International Conference on Evaluation of Novel Approaches to Software Engineering - ENASE, Prague, Czech Republic, INSTICC, SciTePress, 2023, pp. 487–496. doi:10.5220/0011860000003464.
[38] M. Köhler-Bußmeier, H. Rölke, Analysing adaption processes of Hornets, Transactions on Petri Nets and Other Models of Concurrency XVII 14150 (2023). doi:10.1007/978-3-662-68191-6_4.
[39] M. Köhler-Bußmeier, L. Capra, Analysing probabilistic Hornets, in: E. Amparore, L. Mikulski (Eds.), Application and Theory of Petri Nets and Concurrency (PETRI NETS 2025), volume 15714, 2025, pp. 287–309. doi:10.1007/978-3-031-94634-9_14.
[40] M. Köhler-Bußmeier, L. Capra, Simulating self-modifying multi-agent systems with probabilistic nets-within-nets, in: K. Arai (Ed.), Intelligent Systems and Applications, Proceedings of the 2025 Intelligent Systems Conference (IntelliSys), Springer Nature Switzerland, Cham, 2026, pp. 147–165.
[41] M. Köhler, A formal model of multi-agent organisations, Fundamenta Informaticae 79 (2007) 415 – 430.
[42] M. Köhler-Bußmeier, M. Wester-Ebbinghaus, D. Moldt, A formal model for organisational structures behind process-aware information systems, Transactions on Petri Nets and Other Models of Concurrency. Special Issue on Concurrency in Process-Aware Information Systems 5460 (2009) 98–114. doi:10.1007/978-3-642-00899-3_6.