# Exploiting the MAOP Approach for Multi-Level Explainability of Multi-Agent Systems

Elena Yan[1 [0009−0000−6660−9378]], Samuele Burattini[2 [0009−0009−4853−7783]], Luis G. Nardin[1 [0000−0002−4506−2745]], Jomi Fred Hübner[3 [0000−0001−9355−822X]], Olivier Boissier[1 [0000−0002−2956−0533]], Jaime S. Sichman[4 [0000−0001−8924−9643]], and Alessandro Ricci[2 [0000−0002−9222−5092]]

[1] Mines Saint-Etienne, Univ Clermont Auvergne, INP Clermont Auvergne, CNRS, UMR 6158 LIMOS, F-42023 Saint-Etienne France
{elena.yan,gnardin,olivier.boissier}@emse.fr
[2] Alma Mater Studiorum, University of Bologna - Cesena Campus, 47521 Cesena, Italy {samuele.burattini, a.ricci}@unibo.it
[3] Federal University of Santa Catarina, Florianópolis, Brazil jomi.hubner@ufsc.br
[4] Laboratório de Técnicas Inteligentes (LTI), Escola Politécnica (EP), Universidade de São Paulo (USP), São Paulo, Brazil jaime.sichman@usp.br

**Abstract.** Explainability is increasingly becoming an essential non functional requirement for supporting stakeholders to understand complex systems. In multi-agent systems (MAS), we have previously introduced a multi-level explainability framework to explain the behavior of individual agents. In that framework, explainability is investigated from a software engineering perspective and supports stakeholders playing different roles in the software development life cycle (i.e., developer, designer, and end-user). In this paper, we extend that view by moving from an individual agent to a multi-agent system perspective. In particular, we enhance the multi-level explainability framework for MAS by exploiting the benefits of the Multi-Agent Oriented Programming (MAOP) approach. In this view, additional first-class abstractions concerning organization, environment, and interactions introduce a clear separation of concerns in engineering the system and explaining the behavior of MAS.

**Keywords:** Explainable Multi-Agent Systems · Multi-Agent Oriented Programming · Explainability

## 1 Introduction

Explainability, which refers to the ability of a system to be explainable [10], has become an essential non-functional requirement [20,11] of any Artificial Intelligence (AI)-based system [15,24,40].

In Multi-Agent Systems (MAS), *explainable agent* [21] techniques have been proposed to endow agents with the ability to explain their behavior by referring to their internal mental state and perception (e.g., [1,2,13,16,25,33,37]). However, MAS are comprised of a set of agents that perceive and act in an *environment*,

*interact* with each other, and are eventually coordinated and regulated by an *organization* [39]. These various dimensions have been captured in the Multi-Agent Oriented Programming (MAOP) [12,4] approach that defines a set of concepts and associated first-class design and programming abstractions for MAS.

Although Winikoff [36] explores explainability from a multi-agent perspective, explanations are still dependent on individual agents. We argue that focusing only on explaining agents is not enough to achieve system-level explainability in MAS. It has to be enriched to incorporate the explanation of the interaction, environment, and organization dimensions of MAS. To our knowledge, no work has addressed explainability in these dimensions individually or in a global perspective considering their mutual connections.

In this paper, we take a step further towards a comprehensive explanation of MAS by exploiting the MAOP approach to propose an explainability framework that incorporates explanations of the various dimensions of MAS. We extend a *multi-level explainability framework* [41] that presents multiple abstraction levels of explainability (i.e., Implementation, Design, and Domain) by identifying events and narratives to generate explanations concerning all MAOP dimensions. We discuss how these events could be identified and extracted at the Implementation Level based on the JaCaMo [3] platform and mapped to the Design Level based on the MAOP metamodel.

The paper is structured as follows. Section 2 presents the background foundation of our work. Section 3 presents our proposal for (multi-level) explainability of MAS. Section 4 presents the elements to engineer system-level explanations of MAS at multiple levels. Finally, Section 5 concludes with our contributions and future perspectives.

## 2 Background

In this section, we introduce the background concerning our previous multi-level explainability framework (Section 2.1) and the Multi-Agent Oriented Programming approach (Section 2.2) as our adopted metamodel for explaining MAS.

### 2.1 A Multi-Level Explainability Framework

Explainability of MAS can be targeted to different stakeholders [9], such as stakeholders involved in the development of the MAS or end-users. For instance, *developers* require explanations addressing the inner workings and technological aspects of the MAS to support the debugging phase. *Designers* require explanations addressing the agent system's principles and architecture to validate the system requirements. *Domain experts* and *end users* require explanations related to the domain requirements, abstracting from the agent's architecture and technology. We argue that each of these stakeholders would require a different level of abstraction in order to provide explanations using the suitable vocabulary.

A *multi-level explainability framework* (Figure 1) has been proposed in [41] to build explanations of Belief-Desire-Intention (BDI) [6,28] agents at multiple levels of abstraction tailored for the different stakeholders, namely:

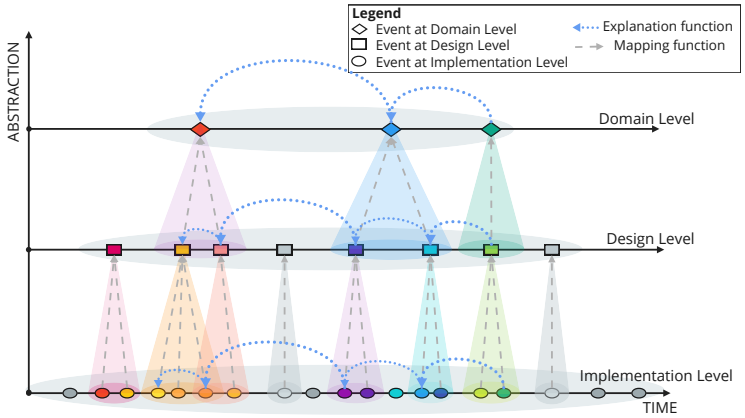


**Fig. 1:** The idea of multiple levels of abstraction for the explanation of a (computing) system, adapted from [41].

- *Implementation Level*, which generates explanations using abstractions and vocabulary related to the technology used to implement MAS (e.g., specific agent-oriented programming language);
- *Design Level*, which generates explanations using abstractions and vocabulary related to the principles and architectures adopted to design the system (e.g., specific cognitive architecture), abstracting from the low-level implementation details; and
- *Domain Level*, which generates explanations using abstractions and vocabulary related to the functional and non-functional requirements of the system as defined with stakeholders and domain experts, dealing with domain-specific knowledge and insights.

The framework generates explanations from execution logs that capture relevant agent events and the semantics underlying the specific level. Events are mapped from lower levels to higher levels and presented to users as *narratives* formed by the description of the events in natural language according to the abstraction of the level. Each level has three main elements to be identified for building explanations: (i) a *set of events* with the associated *narrative* that describes the agent's mental state and behavior based on the abstraction level, (ii) a set of *explanation functions* that defines the causal links among events at the same level of abstraction, and (iii) a *set of mapping functions*, that define the mapping from lower-level events to build higher-level events.

### 2.2 Multi-Agent Oriented Programming Approach

The Multi-Agent Oriented Programming (MAOP) [4,12] approach proposes abstractions structured along different dimensions for the purpose of separation of concerns of engineering and programming MAS (Figure 2).

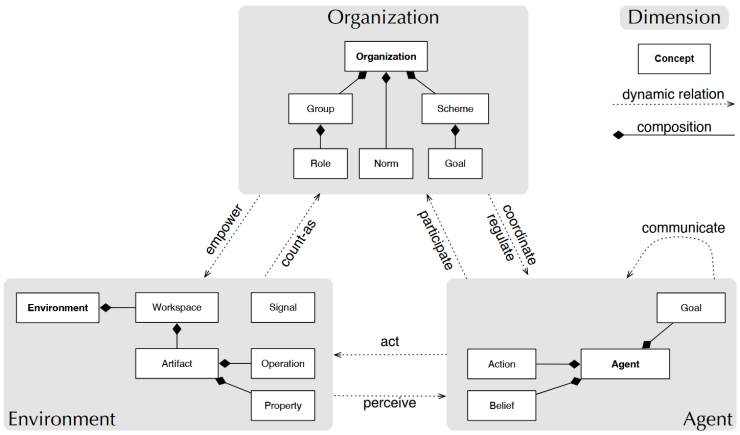


**Fig. 2:** The Agent, Environment, and Organization Dimensions and their concepts in the MAOP paradigm. The Interaction Dimension is given by the connections between the dimensions. Reprinted from [3].

The *Agent Dimension* [34] concerns the mental state and the deliberation of autonomous agents. Agents have a set of *beliefs* and *goals* that they try to achieve by the execution of *plans*. Agents can react to the perceived *events* and act in executing some *actions*.

The *Environment Dimension* [31] concerns the shared space and surrounding conditions in which agents are situated and the shared resources accessible by the agents. The *environment* can be decomposed into one or multiple *workspaces*, which represent the notion of locality and contain *artifacts*. Artifacts are the basic environment and non-autonomous entities that encapsulate computing or other forms of resources that agents can act on, thanks to the set of *operations* and perceive from the *observable properties* and non-persistent *signals*.

The *Organization Dimension* [27] concerns the social structure, composed of groups of agents coordinated with roles and norms to achieve organizational goals. The Organization Dimension includes (i) the structural aspect, defining the *roles*, *links*, and *groups*, (ii) the functional aspect, defining the *missions*, *plans*, and *goals*, and (iii) the normative aspect, defining the *norms* that govern the behavior of agents.

The *Interaction Dimension* [19] concerns the direct and indirect interactions between the Agent, Environment, and Organization dimensions. Agents are able to *communicate* with each other, *perceive* from, and *act* on artifacts. Agents *participate* in organizations, which *regulate* and *coordinate* in return the joint activities of agents. Changes in the state of the environment may also *count as* changes in the state of the organization. Conversely, organizations may *empower* the elements of the environment by allowing them to control and regulate actions or perception of the agents.

## 3 Multi-Level Explainability of Multi-Agent Systems

In this section, we present our global proposal for the explainability of MAS at multiple levels, exploiting the MAOP approach. We use a running example (Section 3.1) to illustrate the explainability at the multiple levels of abstraction (Section 3.2). We conclude the section with some remarks (Section 3.3).

### 3.1 Running Example

The case study that we consider here to exemplify the approach is the *Building House* example, included in the JaCaMo documentation[5].

Expressed in the Domain Level (i.e., from the end-user perspective), Giacomo wants to build a house. For that purpose, he structures his task along two phases:

1. Contracting phase. Giacomo contracts with various specialized companies (Companies are named from A to E) to find and hire the companies in different aspects of the construction and within Giacomo's budget and time limit. His objective here is to hire one company for each of these tasks: prepare the site, lay floors, build walls, build the roof, fit windows and doors, install the plumbing, install the electrical system, and paint the house.
2. Building phase. After the companies have been hired, Giacomo coordinates to execute their tasks on time and in a specific order (some tasks depend on others, and others can be done in parallel).

This scenario is designed as an MAS using the MAOP metamodel and implemented in the JaCaMo platform (part of the code is available in Appendix A). There are two types of *agents* (Agent Dimension): the house *owner* (acting on behalf of Giacomo), who provides the requirements for the house with the budget and time constraints, and *companies* that are characterized by their competencies and offer their service and, if hired, execute the house building tasks.

In the contracting phase, several electronic *auction artifacts* (Environment Dimension) are used to identify the required companies for that task and within the maximum value that the owner can pay for it. The company agents can perceive these artifacts and bid according to their competence and strategies. After some time, Giacomo finishes the auctions, observes the current best bid shown on the artifacts, and hires the winning companies.

When the contracting phase finishes, the building phase starts. A virtual *organization* (Organization Dimension) for the construction of the house is created to assist in the coordination of the execution of the building workflow. In the organization, the groups and roles structure the workflow for executing the tasks, and the norms regulate the roles in executing the tasks as specified.

### 3.2 MAS Explanations at Multiple Levels

In this section, we use the *Building House* example to emphasize the importance of explaining MAS at the Domain, Design, and Implementation Levels.

[5] https://github.com/jacamo-lang/jacamo/tree/main/examples/house-building

We show examples of questions and possible explanations by following the common approach of asking *why* and *why not* questions [23,24,35,37]. We enumerate questions with **Q*** and explanations with **E***. We use the *italic* font in the formulation of questions and explanations to denote a concept that belongs to the vocabulary of the level considered. We use the sans-serif font to denote a specific parameter value of the application.

**Domain Level Explanation** Explanations at the Domain Level deal with concepts in the domain knowledge that are provided by the domain requirements according to a Domain-Driven Design (DDD) [14] methodology. In this level, questions and explanations use the vocabulary defined in the Ubiquitous Language, which is understandable by all stakeholders. Some possible questions and explanations at this level are:

**Q1.** Why was the *company* A not the *winner* for the *task* lay floors? **E1.** The *company* A was not the *winner* for the *task* lay floors because *company* A has a higher *cost*.
**Q2.** Why was the *conclusion* of the *construction* of the house *delayed*? **E2.** The *construction* of the house was *delayed* because the *task* lay floors took longer than *expected*.

**Design Level Explanation** Explanations at the Design Level deal with concepts defined in the adopted MAS design paradigm; in our case, the MAOP metamodel.

Since the MAOP approach offers different dimensions in structuring and providing the separation of concerns in MAS, it is important to address the explainability of each MAOP dimension. Therefore, the stakeholder can ask questions and receive explanations that account for one or more MAOP dimensions. At the Design Level, the vocabulary refers to concepts for agents, interactions, environment, and organization. Some possible questions and explanations at this level, along with the various MAOP dimensions, are:

**Agent Dimension Explanation. Q1.** Why did the *agent* company E *execute* the *action* lay floors? **E1.** The *agent* company E *executed* the *action* lay floors because the *goal* floors laid was *created*.

**Environment Dimension Explanation. Q2.** Why did the *artifact* lay floors auction *update* its *observable property* current bid? **E2.** The *artifact* lay floors auction *updated* its *observable property* current bid because the *operation* bid was *triggered*.

**Organization Dimension Explanation. Q3.** Why did the *organization* not *activate* the *norm* lay floors norm for the *role* bricklayer? **E3.** The *organization* not *activate* the *norm* lay floors norm for the *role* bricklayer because the *organization goal* site prepared was not *achieved* yet.

**Interaction Dimension Explanation. Q4.** Why was a *signal* obligation to achieve floors laid *sent* from the *organization* to the *agent* company E? **E4.**

A *signal* obligation to achieve floors laid was *sent* from the *organization* to the *agent* company E because the *agent* company E *acted* bid on the *artifact* lay floor auction.

Finally, by considering all the MAOP dimensions for explanation, we have richer and wider questions and explanations that can cover all the vocabulary from all the MAOP dimensions, enriching explanations about all aspects of the MAS. For instance, a possible question and explanation are:

**MAOP Dimensions Explanation. Q5.** Why did the *agent* company E *execute* the *action* bid and then *play* the *role* bricklayer? **E5.** The *agent* company E *play* the *role* bricklayer because the *fact* company E is winner *created* from the *artifact* lay floor auction *counts as*[6] *play* the *role* bricklayer.

Let us note that in the previous examples, questions in a single dimension lead to explanations built from the same dimension. However, this is not always the case. The explanation could be built from a single dimension that is different from the dimension used to build the request, or it could be a composition of several dimensions (MAOP Dimensions Explanation).

**Implementation Level Explanation** Explanations at the Implementation Level deal with concepts defined in the adopted MAS platform; in our case, the JaCaMo [3] platform. JaCaMo integrates the Jason [5] platform for programming agents, the CArtAgO [32] platform for programming artifacts, and the Moise [17] platform for programming organizations. For integrating all these platforms, JaCa [31] provides the interface constructs for agents and artifacts, while ORA4MAS [18] provides the implementation of the organization using dedicated organization artifacts. The vocabulary used at this level refers to the (implementation) concepts in JaCaMo platforms and their integration. Some possible questions and explanations at this level, along with the various JaCaMo platforms, are:

**Jason Agent Explanation. Q1.** Why did the *agent* companyA *select* the *plan* +!floors_laid <- layFloors .? **E1.** The *agent* companyA *selected* the *plan* +!floors_laid <- layFloors . because the *goal* floors_laid was *created*.

**CArtAgO Environment Explanation. Q2.** Why was the *observable property* in the *artifact* auction_for_walls currentWinner("no_winner")? **E2.** The *observable property* in the *artifact* auction_for_walls was currentWinner( "no_winner") because there was no *operation triggered* for bid(bidValue).

**Moise Organization Explanation. Q3.** Why was there the *observable property* in the SchemeBoard *organization artifact* bhsch goalState(bhsch, floors_laid,

[6] The count-as is a concept related to constitutive norms, defining which agent, state, or event (i.e., brute fact) in the agents or environment is considered as (i.e., count-as) a particular institutional fact in the organization [7].

[companyE], [companyE], waiting)? **E3.** The *observable property* in the SchemeBoard *organization artifact* bhsch was goalState(bhsch, floors_laid, [companyE], [companyE], waiting) because the *fact* enabled(bhsch, floors_laid) was not *created*.

**JaCaMo Interaction Explanation Q4.** Why was a *message* contract(lay_floors, "hsh_group") *sent* by the *agent* Giacomo to the *agent* companyE? **E4.** The *message* contract(lay_floors, "hsh_group") was *sent* by the *agent* Giacomo to the *agent* companyE because there was a *percept* currentWinneer(companyE) from the *artifact* auction_for_lay_floors to the *agent* Giacomo.

Finally, by considering all the JaCaMo MAS for explanation, we have questions and explanations that can cover all the vocabulary from the JaCaMo platform, enriching explanations about all aspects of the JaCaMo MAS. For instance, a possible question and explanation are:

**JaCaMo MAS Explanation Q5.** Why did the *agent* companyE *achieve* the *goal* floors_laid and there was a *signal* unfulfilled(obligation(companyE, done(bhsch, floors_laid))) *sent* by the NormativeBoard *organization artifact* hsh_group.bhsch? **E5.** The *signal* unfulfilled(obligation(companyE, done(bhsch, floors_laid))) was *sent* by the NormativeBoard *organization artifact* hsh_group.bhsch because the *deadline condition* "2026-02-20" was true and the *agent* companyE *executed* the *goal* floors_laid in "2026-02-25".

### 3.3 Remarks

We provided examples to illustrate explanations in different dimensions of the MAS concerning the agent, environment, organization, and interaction at multiple levels. The MAOP metamodel offers a separation of concerns for programming MAS that can be exploited in explaining, enabling the *separation of concerns in explainability*. The stakeholder is thus able to focus on the explanation of particular dimensions.

Explainability of MAS can be considered at multiple levels of abstraction as well. Different levels of abstractions use different vocabulary, which can be achieved thanks to the mapping functions that link the vocabulary for generating narratives from a lower level to a higher level. For instance, it is intuitive to think that the (CArtAgO) *artifact* auction_for_lay_floors (Implementation Level) can be mapped to *artifact* lay floors auction in the Environment Dimension (Design Level) and to the concept *auction* lay floors (Domain Level).

Because the Agent Dimension explainability has already been tackled in the previous work and in the existing literature, in the next sections, we will discuss how to build explanations concerning the environment, organization, and interaction at multiple levels.

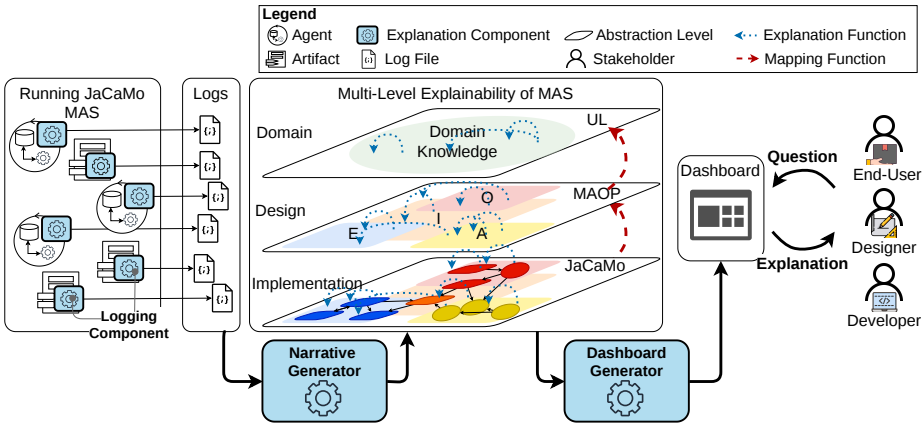


**Fig. 3:** Global architecture of the multi-level explainability framework for MAS, explanations and narratives at the Domain Level use vocabulary from the Ubiquitous Language (UL), the Design Level vocabulary from the MAOP approach, and the Implementation Level vocabulary from the JaCaMo framework.

## 4 A Framework for Engineering MAS Explanations at Multiple Levels

In this section, we describe the building blocks for engineering the multi-level explainability framework for MAS. We present first the global architecture (Section 4.1), and then the elements for engineering explanations for the Implementation Level (Section 4.2) and Design Level (Section 4.3).

### 4.1 Global Architecture

Figure 3 shows the proposed global architecture for a multi-level explainability of MAS based on the JaCaMo platform.

We adopt the logging approach as a way to collect information about the running JaCaMo MAS. A *logging component* can be integrated into each JaCaMo MAS agent or artifact to log all the relevant events in a log file. This component encapsulates the algorithms for selecting and saving the events generated by the entity in a log file. In the JaCaMo platform, we are able to customize the agent architecture (as done in [41]) and extend the artifact class to enable collecting logs about all the MAS.

All the collected logs are then used to build events and narratives. A *narrative generator* component is responsible for processing all the logs and generating narratives at different levels of abstraction. This component encapsulates the algorithms for implementing the explanation function and mapping functions. As mentioned, narratives at the Implementation Level are expressed using the vocabulary from the JaCaMo platform, narratives at the Design Level are ex-

pressed using the vocabulary from the MAOP Approach, and narratives at the Domain Level are expressed using vocabulary from the Ubiquitous Language.

The last component is the *narrative dashboard*, which presents an interface for the stakeholders to interact with. Different types of stakeholders (e.g., developer, designer, and end-user) can select the desired level of abstraction in the interface and ask questions and receive explanations about the running MAS.

### 4.2 The Implementation Level

To build explanations, we need to identify the events with their narratives and the explanation function. There is no mapping function at this level because events are extracted and narratives are generated directly from the logs. We use the Building House example (Section 3.1) to illustrate the several elements.

**Events and Narratives** Here we illustrate the set of the main events and their associated narratives of the CArtAgO Environment, Moise Organization, and the JaCaMo Interaction at the Implementation Level. See Appendix B for the detailed set of events and narratives.

*CArtAgO Environment.* Integrating the logging component to CArtAgO artifacts enables us to collect the relevant events of the artifacts. Table 1 presents the set of events with their narrative templates[7] concerning the updates of observable properties, the lifecycle of operations (operation triggered, executed, or failed), and signals that are sent. In the CArtAgO Environment, there are present two types of artifacts: *domain artifacts*, i.e., artifacts dedicated to the management of the domain knowledge and problems, and *environment artifacts*, i.e., artifacts dedicated to the management of the environment. For instance, AuctionArtifact is a domain artifact that exposes the operation bid. When company agents execute this action, the operation is triggered and executed:

1. Operation Triggered. Operation bid triggered with parameters 2000 in the artifact auction_for_site_preparation.
2. Operation Executed. Operation bid executed with parameters 2000 in the artifact auction_for_site_preparation.

The CArtAgO Environment provides a set of default environment artifacts that enable agents to manage workspaces in a (distributed) node (i.e., NodeArtifact) and the artifacts in a workspace (i.e., WorkspaceArtifact). The NodeArtifact exposes operations to create, join, and leave a workspace. The WorkspaceArtifact exposes operations to make, focus, and dispose of an artifact. The logging component in these environment artifacts can log the events based on the template in the previous Table 1 (examples in Appendix B.1).

[7] Each narrative is expressed as a template containing monospaced placeholders (e.g., `[opName]`) that are instantiated with concrete information expressed in sans-serif font (e.g., createWorkspace) when the event is generated at execution time. In the tables, horizontal separators group events related to the same concept.

| Event | Narrative |
|---|---|
| Observable Property Updated | Observable property `[propName]` updated to `[propValue]` in the artifact `[artName]` |
| Operation Triggered | Operation `[opName]` triggered with parameters `[opParam]` in the artifact `[artName]` |
| Operation Executed | Operation `[opName]` executed with parameters `[opParam]` and return value `[returnValue]` in the artifact `[artName]` |
| Operation Failed | Operation `[opName]` failed with parameters `[opParam]` and reason `[reason]` in the artifact `[artName]` |
| Signal Sent | Signal `[signal]` sent by the artifact `[artName]` |

**Table 1:** Set of events and narratives related to CArtAgO Environment at the Implementation Level.

*Moise Organization.* The Moise Organization is implemented in ORA4MAS [18] by using dedicated *organization artifacts* to manage the organization, defined by an organization specification. An OrgBoard organization artifact that manages all the organization artifacts representing the group, scheme, and normative board of the organization. The GroupBoard organization artifact exposes operations to manage a role (add, adopt, or leave) and a scheme that the group is responsible for (add or remove). The SchemeBoard organization artifact exposes operations to indicate the status of a mission (committed or left) and a goal (achieved, failed, or released). Lastly, the NormativeBoard organization artifact manages the dynamics of norms. Norms are computed based on the current facts that occur in the system, therefore it exposes operations to add and remove a fact and signals to send the norms. Whenever the state of the norms is updated (i.e., active, fulfilled, unfulfilled, and inactive), a signal is sent by the NormativeBoard organization artifact. Events concerning the Moise Organization at the Implementation Level are generated by those organization artifacts using the basic set of events presented in Table 1 (examples in Appendix B.2).

*JaCaMo Interaction.* The JaCaMo Interaction concerns events that enable the connection with the agents, environment, and organization. Because the JaCaMo MAS is implemented by agents and artifacts, the basic primitive events concerning the interaction are (Table 2): (i) messages from agents to agents, (ii) actions from agents to artifacts, (iii) perceptions from artifacts to agents, and (iv) signals from artifacts to agents. For instance, when the goal floors_laid has to be executed by the agent companyE playing the role of bricklayer, the following event regarding a signal is sent by the NormativeBoard organization artifact:

> Signal Sent. Signal active(obligation(companyE, enabled(bhsch, floors_laid), done(bhsch, floors_laid, companyE), '20 minutes')) sent by the artifact NormativeBoard.

| Event | Narrative |
|---|---|
| New Message | New message of type `[performative]` from the agent `[agName1]` sent with the content `[content]` to the agent `[agName2]` |
| New Action | New action `[action]` from `[agName]` to `[artName]` |
| New Percept | New percept `[content]` from `[artName]` to `[agName]` |
| New Signal | New Signal sent from the artifact `[artName]` with content `[content]` |

**Table 2:** Set of events and narratives related to the JaCaMo Interaction at the Implementation Level.

**Explanation Function** Having identified the events produced during the MAS execution, the explanation function establishes the causal relationship among them, linking each event to the past event(s) that it *is explained by*.

There are different ways to design the explanation function, for instance, we can explain an event with the previous event from the same artifact: an "Operation Executed" is explained by a previous "Operation Executed". For instance, the Operation addRole executed (last event in the sequence) *is explained by* (←) a previous event operation createScheme executed (details are omitted with [...]):

Operation createScheme executed [...] ← Operation addRole executed [...].

Another way is to explain the events referring to the causal chain from events generated by other entities. For instance, the "Operation Executed" event *is explained by* a previous "New Action" in the JaCaMo Interaction and "External Action Triggered"[8] in the Jason Agent.

[...] ← External action bid triggered [...] ← New action bid [...] ← Operation bid triggered [...] ← Operation bid executed [...].

Note that when linking events generated from different entities (that can be potentially distributed on different machines), a challenge is to align the events based on correlated information and the time that is logged.

### 4.3 The Design Level

The explanation at the Design Level follows the concepts in the MAOP metamodel. We show in this section examples of events and their narratives, explanation functions linking the events, and mapping functions from the Implementation Level to build explanations at the Design Level. We use the Building House example (Section 3.1) to illustrate the several elements. See Appendix C for the detailed set of events and narratives.

[8] The "External Action Triggered" event in the Jason Agent can be further explained by a "Goal Created" event, see details in [41].

**Events and Narratives** Here, we describe the set of events and narratives for the Environment, Organization, and Interaction Dimensions at the Design Level following the MAOP metamodel.

*Environment Dimension.* Table 3 presents the events for managing workspaces in a node, artifacts in a workspace, and events related to the state of an artifact. Note that, differing from the Implementation Level, concepts related to workspace and artifact are represented as first-class abstractions in the events and narratives and abstracted from implementation details. For instance, the events of "Artifact Created" and "Operation Executed" at the Design Level are:

1. Artifact Created. Artifact auction_for_lay_floor created.
2. Operation Executed. Operation bid executed in auction_for_lay_floor.

| **Event** | **Narrative** |
|---|---|
| Workspace Created | Workspace `[wspName]` created |
| Artifact Created | Artifact `[artName]` created in workspace `[wspName]` |
| Artifact Disposed | Artifact `[artName]` disposed in workspace `[wspName]` |
| Observable Property Updated | Observable Property `[propName]` updated to `[propValue]` in `[artName]` |
| Operation Executed | Operation `[opName]` executed in `[artName]` |
| Operation Failed | Operation `[opName]` failed in `[artName]` |

**Table 3:** Set of events and narratives related to the Environment Dimension at the Design Level.

*Organization Dimension.* Events and narratives of the Organization Dimension at the Design Level have as first-class abstractions concepts concerning groups, roles, social schemes, goals and missions in a scheme, and norms. Events record the lifecycle of a group (created and deleted), role in a group (adopted and left), scheme in a group (created, assigned, finished, disconnected, and deleted), mission in a scheme (committed and left), goal in a scheme (waiting, enabled, achieved, impossible, and removed), and norm for a role to achieve a goal or mission (activated, fulfilled, unfulfilled, and inactivated). Examples of events and narratives are presented in Table 4 (details in Appendix C.2). For instance, narratives about the creation of the group responsible for building the house are:

1. Group Created. Group house_group created.
2. Role Adopted. Role house_owner adopted by Giacomo.

   ...
9. Scheme Assigned. Scheme bhsch assigned to group house_group.
10. Mission Committed. Mission management_of_house_building committed by house_owner.

| Event | Narrative |
|---|---|
| Group Created | Group `[groupName]` created |
| Group Deleted | Group `[groupName]` deleted |
| Scheme Created | Scheme `[schemeName]` is created |
| Scheme Finished | Scheme `[schemeName]` is finished |
| Role Adopted | Role `[roleName]` adopted by `[agName]` |
| Role Left | Role `[roleName]` left by `[agName]` |
| Mission Committed | Mission `[missionName]` committed by `[agName]` |
| Mission Removed | Mission `[missionName]` removed |
| Norm Activated | Norm `[norm]` activated for the agent `[agName]` |
| Norm Fulfilled | Norm `[norm]` fulfilled by the agent `[agName]` |
| Norm Unfulfilled | Norm `[norm]` unfulfilled by the agent `[agName]` |

**Table 4:** Main events with their narratives at Design Level related to the Organization Dimension (full Table in Appendix C.2).

*Interaction Dimension.* Events in the Interaction Dimension concern the connection between the Agent, Environment, and Organization Dimensions. Besides the basic primitives to interact with agents and artifacts, there are also events for the agents to interact in a workspace and in a group (Table 5). Agents in the workspace can focus on, perceive from, and act on the artifacts. Agents in a group can adopt or leave a role, commit to the missions and goals in accordance with the progress of the scheme execution.

**Explanation Function** The explanation function at the Design Level can refer to the high-level semantics of the MAOP approach. Different explanation functions can be designed. For instance, a possible explanation function is to link the events to the first event of their lifecycle. Events in a workspace are explained by the event "Workspace Created". Events related to an artifact are explained by the event "Artifact Created". For instance, the companyE agent focusing on the auction_for_lay_floor artifact *is explained by* the event "Artifact Created":

> Artifact auction_for_lay_floor created [...] ← Agent companyE focused artifact auction_for_lay_floor [...].

Events related to a group are explained by the event "Group Created". Events related to a scheme, goals, and norms for the scheme are explained by the event "Scheme Created". For instance, the norm to achieve floors_laid for the agent companyE that plays bricklayer *is explained by* the event "Scheme Created":

> Scheme bhsch created ← Norm (obligation, bricklayer, floors_laid) activated for the agent companyE.

| Event | Narrative |
|---|---|
| Agent Communicated Agent | Agent `[agName1]` communicated `[content]` to the agent `[agName2]` |
| Agent Joined Workspace | Agent `[agName]` joined the workspace `[wspName]` |
| Agent Focused Artifact | Agent `[agName]` focused on artifact `[artName]` in the workspace `[wspName]` |
| Agent Acted Operation | Agent `[agName]` acted on operation `[opName]` of `[artName]` |
| Agent Perceived Property | Agent `[agName]` perceived property `[propName]` from `[artName]` |
| Agent Joined Group | Agent `[agName]` joined the group `[groupName]` |
| Agent Adopted Role | Agent `[agName]` adopted the role `[roleName]` |
| Agent Achieved Goal | Agent `[agName]` achieved the goal `[goalName]` |
| Agent Gave Up Goal | Agent `[agName]` gave up the goal `[goalName]` |

**Table 5:** Main events with their narratives in the Interaction Dimension at the Design Level (full Table in Appendix C.3).

**Mapping Functions** The mapping functions describe how events at the Implementation Level are *mapped to* ($\Rightarrow$) events at the Design Level. Possible mapping functions are described here (details in Appendix D).

Events generated from environment artifacts and domain artifacts are mapped to the events in the Environment Dimension. For instance:

> ⟨Operation Triggered, Operation Executed⟩ with `[opName]` = makeArtifact and `[artName]` = WorkspaceArtifact ⇒ Artifact Created with `[artName]` = `[opParam]`.[9]

Similarly, events generated from organization artifacts are mapped to the events in the Organization Dimension, while events related to the JaCaMo Interaction connect the events from the Jason agents, CArtAgO Environment, and Moise Organization. For instance:

> ⟨Observable Property Updated, New Percept, Belief Updated⟩ with `[artName]` = `[artName]`, `[agName]` = `[agName]` ⇒ Agent Perceived Property.

## 5 Conclusion and Future Work

In this paper, we expand the scope of explainability from individual agents to the whole MAS, exploiting the MAOP metamodel for engineering the explainability of the Environment, Organization, and Interaction Dimensions. We extend

[9] The mapping can be read as: the sequence of Implementation Level events "Operation Triggered" and "Operation Executed" with operation `[opName]` equals to makeArtifact and `[artName]` equals to WorkspaceArtifact *is mapped to* the Design Level event "Artifact Created" with parameter `[artName]` equals to `[opParam]` of the previous "Operation Executed" event.

our previous work [41] by presenting MAS explanations at different levels. We discussed the building blocks for engineering the framework and provided examples for the Implementation and Design Levels. The explanation functions can be designed in various ways and evaluated accordingly (as for agents [16,38]). The mapping functions link from the Implementation Level to the Design Level. This modular approach enables reaching explanations at the Domain Level for end-users. The Domain Level and the concrete implementation of the work are ongoing, with the addition of the required explainability components to build the explanations for MAS.

We have defined only three levels, but other levels can be envisioned. For instance, an intermediary level between the Design and Domain Levels, combining the MAOP metamodel with the Domain Level. We advocate that the Domain Level, even if larger than the MAOP metamodel, could get inspirations along similar dimensions that are quite natural in human societies and be enriched with methodologies from the Domain Driven Design (DDD) (as for bringing agents in DDD [29]) and works in the Knowledge Level [26].

Although our current focus is on the JaCaMo MAS platform, we envision that the framework can be applied to other MAS platforms that adopt (or not) the MAOP metamodel. In that case, it needs to adapt the events of the Implementation Level and the mapping to the Design Level. The same events and narratives at the Design Level and the mapping to the Domain Level can be fully reused. We envision that, we can further exploit the Design Level and the MAOP metamodel to provide richer explanations in terms of the MAOP dimensions for MAS that do not adopt the MAOP metamodel (e.g., Multi-Agent Reinforcement Learning (MARL) [8] or Large Language Model (LLM)-based MAS [22]). In that line, the Design Level could act as a unified and intermediary neck for MAS (as for the cognitive hourglass [30] for agents).

The different examples used at the Design Level have shown the interest as a future direction of research to use the MAOP metamodel and approach for structuring and defining a rich set of explanations along the four dimensions taken in separation or all together, bringing the *separation of concerns in explainability*. On one hand, stakeholders can select which dimension to focus on and receive explanations, and on the other hand, it supports designers of the explainability framework in structuring and developing different parts and forms of explanations that can be generated.

**Acknowledgments.** This study is partially funded by ANR-FAPESP NAIMAN project (ANR-22-CE23-0018-01, FAPESP 2022/03454-1) and "European Lighthouse to Manifest Trustworthy and Green AI" (ENFIELD) from the European Union's Horizon Europe research and innovation program under grant agreement No. 101120657. The authors are also members of the UNBIAS team, which is a component of the THUS pillar of the USP-CNRS International Research Center.

## References

1. Anjomshoae, S., Najjar, A., Calvaresi, D., Främling, K.: Explainable agents and robots: Results from a systematic literature review. In: Proceedings of the 18th

International Conference on Autonomous Agents and MultiAgent Systems. pp. 1078—-1088. AAMAS '19, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2019)
2. Beaumont, K., Yan, E., Burattini, S., Collier, R.: Engineering inter-agent explainability in BDI agents. In: International Workshop on EXplainable, Trustworthy, and Responsible AI and Multi-Agent Systems - International Workshop, EXTRAAMAS 2025, Detroit, Michigan, USA, May 20, 2025 (2025)
3. Boissier, O., Bordini, R.H., Hubner, J., Ricci, A.: Multi-agent oriented programming: programming multi-agent systems using JaCaMo. MIT Press (2020)
4. Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A.: Dimensions in programming multi-agent systems. The Knowledge Engineering Review **34**, e2 (2019). https://doi.org/10.1017/S026988891800005X
5. Bordini, R., Hübner, J., Wooldridge, M.: Programming Multi-Agent Systems in AgentSpeak Using Jason, vol. 8. John Wiley & Sons, Hoboken, NJ (10 2007). https://doi.org/10.1002/9780470061848
6. Bratman, M.: Intention, Plans, and Practical Reason. Cambridge, MA: Harvard University Press, Cambridge (1987)
7. de Brito, M., Hübner, J.F., Boissier, O.: Coupling the normative regulation with the constitutive state management in situated artificial institutions. Knowledge Engineering Review **34**, e21 (2019). https://doi.org/10.1017/S026988891900016X
8. Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **38**(2), 156–172 (2008). https://doi.org/10.1109/TSMCC.2007.913919
9. Caglar, T., Sreedharan, S., Vered, M.: Who am I dealing with? Explaining the designer's hidden intentions. In: Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems. pp. 436—-444. AAMAS '25, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2025)
10. Chazette, L., Brunotte, W., Speith, T.: Exploring explainability: A definition, a model, and a knowledge catalogue. In: 2021 IEEE 29th International Requirements Engineering Conference (RE). pp. 197–208. IEEE (2021). https://doi.org/10.1109/RE51729.2021.00025
11. Chazette, L., Karras, O., Schneider, K.: Do end-users want explanations? analyzing the role of explainability as an emerging aspect of non-functional requirements. In: 2019 IEEE 27th International Requirements Engineering Conference (RE). pp. 223–233 (2019). https://doi.org/10.1109/RE.2019.00032
12. Demazeau, Y.: Steps towards multi-agent oriented programming. In: First International Workshop on Multi Agent Systems. Boston, MA (1997)
13. Dennis, L.A., Oren, N.: Explaining BDI agent behaviour through dialogue. In: Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems. pp. 429—437. AAMAS '21, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2021). https://doi.org/10.1007/s10458-022-09556-8
14. Evans, E.: Domain-Driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley (2004)
15. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5) (Aug 2018). https://doi.org/10.1145/3236009

16. Harbers, M., van den Bosch, K., Meyer, J.J.: Design and evaluation of explainable BDI agents. In: Huang, J.X., Ghorbani, A.A., Hacid, M., Yamaguchi, T. (eds.) Proceedings of the 2010 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT 2010, Toronto, Canada, August 31 - September 3, 2010. pp. 125–132. IEEE Computer Society Press, Toronto, Canada (2010). https://doi.org/10.1109/WI-IAT.2010.115
17. Hubner, J.F., Sichman, J.S., Boissier, O.: Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels. Int. J. Agent-Oriented Softw. Eng. **1**(3/4), 370—-395 (Dec 2007). https://doi.org/10.1504/IJAOSE.2007.016266
18. Hübner, J.F., Boissier, O., Kitio, R., Ricci, A.: Instrumenting multi-agent organisations with organisational artifacts and agents. Autonomous Agents and Multi Agent Systems **20**(3), 369–400 (2010). https://doi.org/10.1007/S10458-009-9084-Y
19. Huhns, M.N.: Interaction-oriented programming. In: Ciancarini, P., Wooldridge, M.J. (eds.) Agent-Oriented Software Engineering, First International Workshop, AOSE 2000, Limerick, Ireland, June 10, 2000, Revised Papers. Lecture Notes in Computer Science, vol. 1957, pp. 29–44. Springer (2001). https://doi.org/10.1007/3-540-44564-1_2
20. Köhl, M.A., Baum, K., Langer, M., Oster, D., Speith, T., Bohlender, D.: Explainability as a non-functional requirement. In: 2019 IEEE 27th International Requirements Engineering Conference (RE). pp. 363–368 (2019). https://doi.org/10.1109/RE.2019.00046
21. Langley, P., Meadows, B., Sridharan, M., Choi, D.: Explainable Agency for Intelligent Autonomous Systems. In: Singh, S., Markovitch, S. (eds.) Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017. pp. 4762–4764. AAAI Press, San Francisco, California, USA (2017). https://doi.org/10.1609/aaai.v31i2.19108
22. Li, X., Wang, S., Zeng, S., Wu, Y., Yang, Y.: A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges. Vicinagearth **1**(1) (Oct 2024). https://doi.org/10.1007/s44336-024-00009-2
23. Lim, B.Y., Dey, A.K., Avrahami, D.: Why and why not explanations improve the intelligibility of context-aware intelligent systems. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. pp. 2119—-2128. CHI '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1518701.1519023
24. Miller, T.: Explanation in artificial intelligence: Insights from the social sciences. Artificial Intelligence **267**, 1–38 (2019). https://doi.org/10.1016/j.artint.2018.07.007
25. Morveli Espinoza, M., Possebom, A.T., Tacla, C.A.: Argumentation-based agents that explain their decisions. In: 2019 8th Brazilian Conference on Intelligent Systems (BRACIS). pp. 467–472 (2019). https://doi.org/10.1109/BRACIS.2019.00088
26. Newell, A.: The knowledge level. Artificial Intelligence **18**(1), 87–127 (1982). https://doi.org/10.1016/0004-3702(82)90012-1
27. Pynadath, D.V., Tambe, M., Chauvat, N., Cavedon, L.: Toward team-oriented programming. In: Jennings, N.R., Lespérance, Y. (eds.) Intelligent Agents VI, Agent Theories, Architectures, and Languages (ATAL), 6th International Workshop, ATAL '99, Orlando, Florida, USA, July 15-17, 1999, Proceedings. Lecture Notes in Computer Science, vol. 1757, pp. 233–247. Springer (1999). https://doi.org/10.1007/10719619_17

28. Rao, A.S., Georgeff, M.P.: BDI agents: From theory to practice. In: Lesser, V.R., Gasser, L. (eds.) Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA. pp. 312–319. The MIT Press (1995)
29. Ricci, A., Burattini, S., Ciortea, A., Castellucci, M.: Agents for ddd – back and forth. In: Engineering Multi-Agent Systems: 12th International Workshop, EMAS 2024, Auckland, New Zealand, May 6–7, 2024, Revised Selected Papers. p. 175–188. Springer-Verlag, Berlin, Heidelberg (2024). https://doi.org/10.1007/978-3-031-71152-7_11
30. Ricci, A., Mariani, S., Zambonelli, F., Burattini, S., Castelfranchi, C.: The cognitive hourglass: Agent abstractions in the large models era. In: Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems. pp. 2706—-2711. AAMAS '24, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2024)
31. Ricci, A., Piunti, M., Viroli, M.: Environment programming in multi-agent systems: an artifact-based perspective. Autonomous Agents and Multiagent Systems. **23**(2), 158–192 (2011). https://doi.org/10.1007/S10458-010-9140-7
32. Ricci, A., Viroli, M., Omicini, A.: Programming MAS with artifacts. In: Bordini, R.H., Dastani, M.M., Dix, J., El Fallah Seghrouchni, A. (eds.) Programming Multi-Agent Systems. pp. 206–221. Springer Berlin Heidelberg, Berlin, Heidelberg (2006). https://doi.org/10.1007/11678823_13
33. Rodriguez, S., Thangarajah, J., Davey, A.: Design patterns for explainable agents (XAg). In: Dastani, M., Sichman, J.S., Alechina, N., Dignum, V. (eds.) Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024, Auckland, New Zealand, May 6-10, 2024. pp. 1621–1629. ACM, Richland, SC (2024)
34. Shoham, Y.: Agent-oriented programming. Artificial Intelligence **60**(1), 51–92 (1993). https://doi.org/10.1016/0004-3702(93)90034-9
35. Winikoff, M.: Debugging agent programs with why? questions. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems. pp. 251––259. AAMAS '17, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2017)
36. Winikoff, M.: Towards Engineering Explainable Autonomous Systems. In: Briola, D., Cardoso, R.C., Logan, B. (eds.) Engineering Multi-Agent Systems - 12th International Workshop, EMAS 2024, Auckland, New Zealand, May 6-7, 2024, Revised Selected Papers. Lecture Notes in Computer Science, vol. 15152, pp. 144–155. Springer, Cham (2024). https://doi.org/10.1007/978-3-031-71152-7_9
37. Winikoff, M., Sidorenko, G., Dignum, V., Dignum, F.: Why bad coffee? Explaining BDI agent behaviour with valuings. Artificial Intelligence **300**, 103554 (2021). https://doi.org/10.1016/j.artint.2021.103554
38. Winikoff, M., Thangarajah, J., Rodriguez, S.: A scoresheet for explainable AI. In: Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems. pp. 2171—2180. AAMAS '25, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2025)
39. Wooldridge, M.: An introduction to multiagent systems. John wiley & sons, USA (2009)
40. Xu, F., Uszkoreit, H., Du, Y., Fan, W., Zhao, D., Zhu, J.: Explainable AI: A brief survey on history, research areas, approaches and challenges. In: Tang, J., Kan, M.Y., Zhao, D., Li, S., Zan, H. (eds.) Natural Language Processing and Chinese Computing. pp. 563–574. Springer International Publishing, Cham (2019). https://doi.org/10.1007/978-3-030-32236-6_51

41. Yan, E., Burattini, S., Hübner, J.F., Ricci, A.: A multi-level explainability framework for engineering and understanding BDI agents. Autonomous Agents and Multiagent Systems. **39**(1), 9 (2025). https://doi.org/10.1007/S10458-025-09689-6