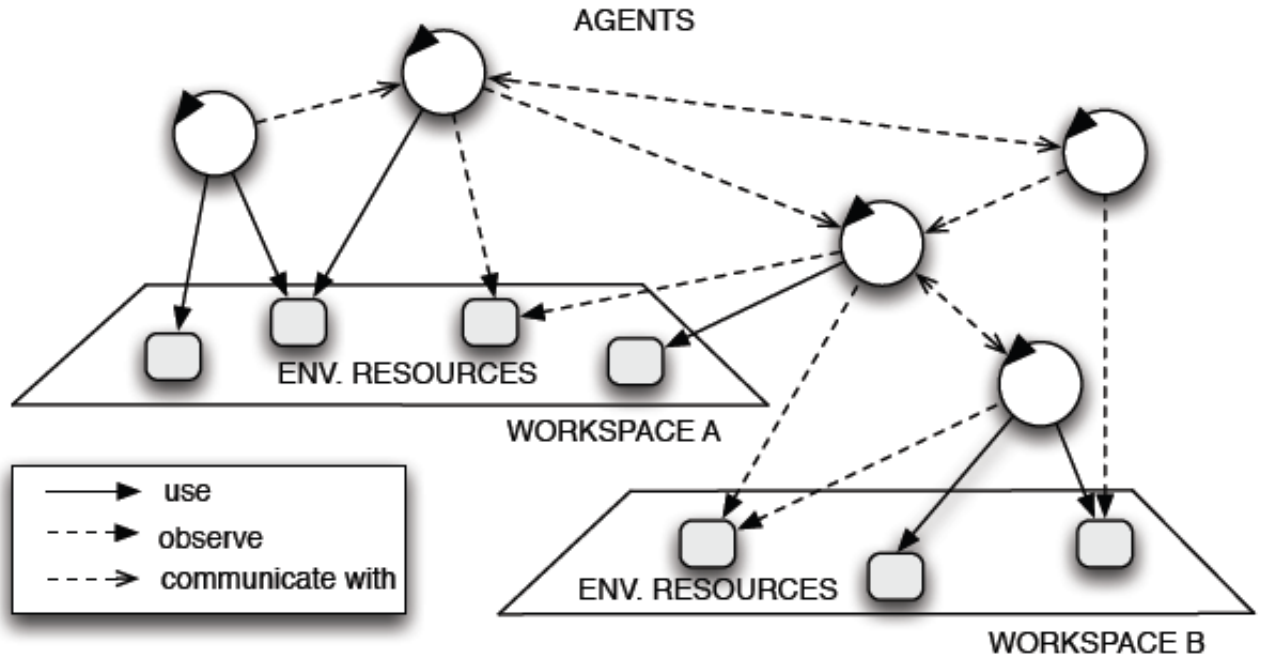
The JaCaMo approach

The **JaCaMo** framework is rooted on a specific programming model named **JaCa**. Therefore, before describing in details how to use **JaCaMo** for realising agent-based application we briefly introduce the **JaCa** programming model.

The **JaCa** programming model

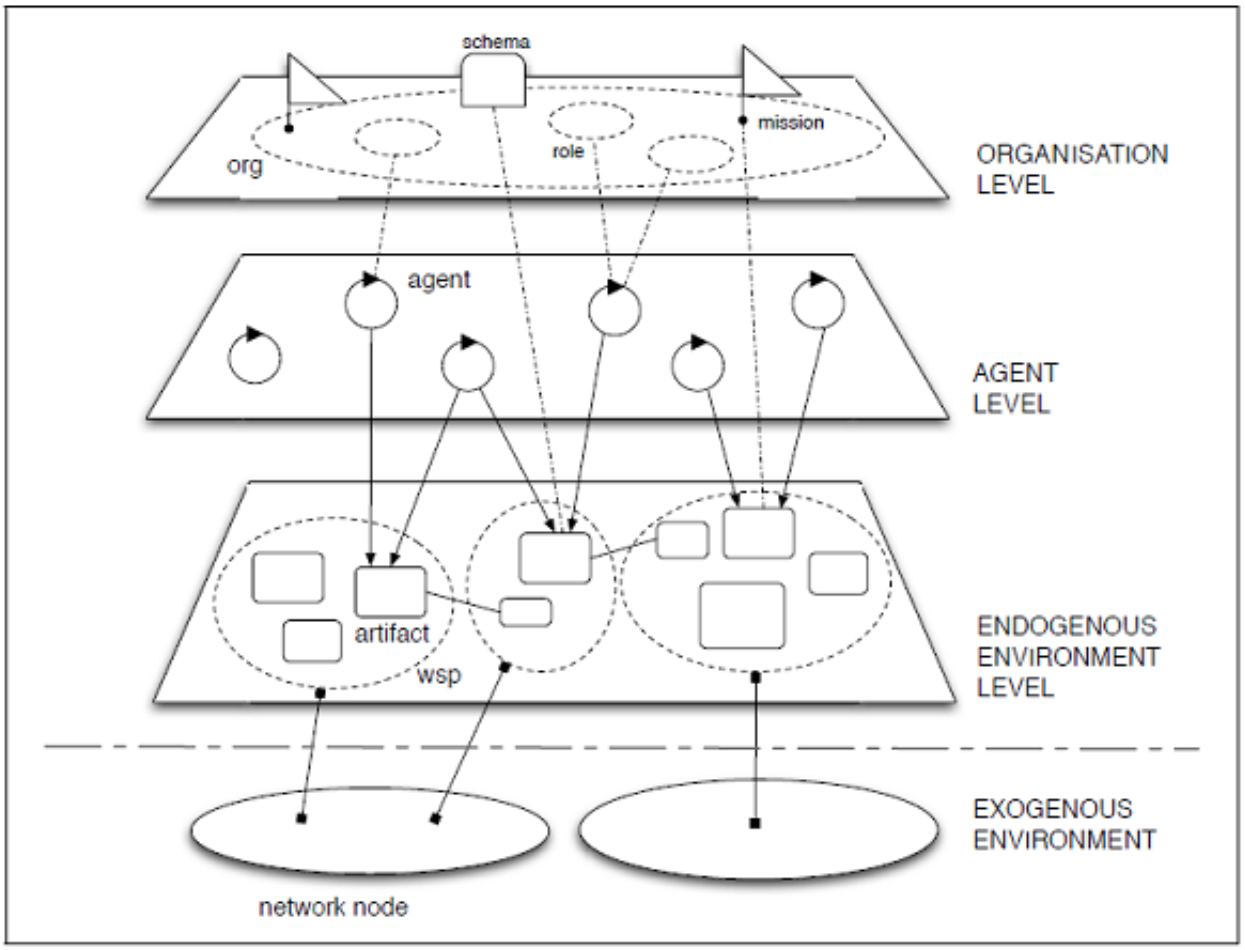
In **JaCa** a MAS is designed and programmed as a set of agents which work and cooperate inside a common environment. Programming the application means then programming the agents on the one side, encapsulating the logic of control of the tasks that must be executed, and the environment on the other side, as a first-class abstraction providing the actions and functionalities exploited by agents to do their tasks. It is worth remarking that this is an**endogenous** notion of environment, i.e. the environment here is part of the software system to be developed [1].

More specifically, in **JaCa** Jason [2] is adopted as programming language to implement and execute the agents and **CArtAgo** [3] as the framework to program and execute the environments.



The **JaCaMo** approach

A **JaCaMo** multi-agent system or, equivalently, a software system programmed in **JaCaMo** is given by a Moise organisation of autonomous BDI agents, programmed in Jason, working in shared distributed artifact-based environments, programmed in **CArtAgO** (see figure below).



Each of the three independent platforms composing the **JaCaMo** framework has its own set of programming abstractions and its reference programming model and meta-model. Therefore, for the **JaCaMo** framework we have considered as keystone the definition of a global programming meta-model that takes into account all the abstractions made available by each platform.

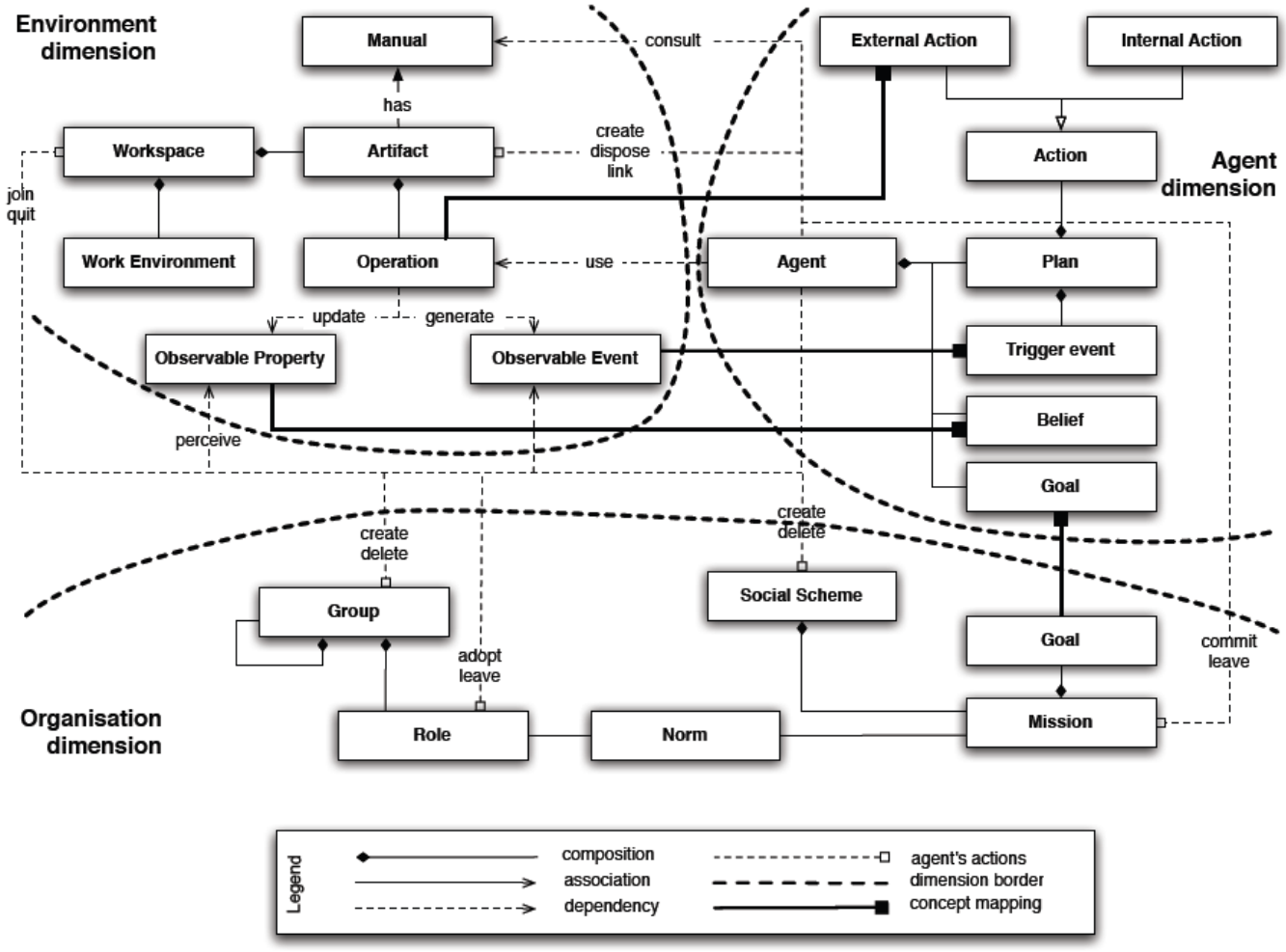
This meta-model has been realised with the aim to define the dependencies, the connections and — more importantly — the conceptual mappings and the synergies between all the different abstractions available in the three platforms programming meta-models. The starting points for realising such an integrated meta-model are represented by other work in which we have already started to explore the synergy between: *(i)* the agent and the environment dimension [1], *(ii)* the agent and organisation dimension [4], and *(iii)* the organisation and environment dimension [5].

The abstractions belonging to the agent dimension, related to the **Jason** meta-model, are mainly inspired by the BDI architecture upon which **Jason** is rooted. So an agent is an entity composed of a set of *beliefs*, representing agent's current state and knowledge about the environment in which it is situated, a set of *goals*, which correspond to tasks the agent has to perform/achieve, and a set of *plans* which are courses of actions, either internal or external, triggered by events, and that agents can dynamically compose, instantiate and execute to

achieve goals.

For the environment side, each **CArtAgO** environment instance — the *work environment*entity in the picture — is composed by one or more *workspace* entities. Each workspace is formed by a certain set of *artifacts* which provide a set of *operations* and *observable properties*[6] defining an artifact's usage interface. Operation execution could generate updates to the observable properties and specific *observable events*. The last entity concerning the environment dimension is the *manual*, an entity used for representing the description of the functionalities provided by an artifact.

Finally, in regards to the **Moise** organisational meta-model we have that: *(i)* the structural specification is described by the *group* and *role* entities — both defining the structure of the different agents groups and sub-groups within the organisation; *(ii)* the functional specification is defined by *social scheme*, *mission*, and *goal* entities — the social scheme defines the structure of the organisation's goals (structured as missions); and finally *(iii)* the normative specification is defined through the *norm* entity which binds roles to missions, constraining the agent's behavior when it enters a group playing a certain role.



(Click the image for a larger version)

Synergies between the programming dimensions in JaCaMo

In the meta-model picture, the synergies among the three programming dimensions are represented by connections terminating with a filled or not filled square. The connections terminating with a filled square are the most important part of our integrated meta-model as they explicitly represent the synergies and the conceptual mappings we have identified during the definition of our integrated approach: these connections provide for free the integration between the different dimensions, an integration that in other approaches must be programmed by users in ad *ad hoc* manner.

The synergy between the Agent and Environment (A-E) dimensions is based on the **A&A**meta-model [6]. As detailed in [6], from an agent perspective, interactions with artifacts and workspaces are defined according to a rigorous semantics based on actions and percepts. By joining and working in a workspace, the repertoire of an agent's actions is dynamic and is given by the overall set of operations provided by the dynamic set of artifacts currently available in that workspace — this is the meaning of the connection in figure between the agent's external actions and the artifact's operations. So, for example, an agent may perform a *bid* action since in the workspace there is a shared instance of an *auction artifact* providing that operation.

On the perception side, artifact observable properties and events are mapped into agent percepts, and so, for BDI agents for instance, the dynamic observable state of an artifact — given by the set of observable properties — is mapped, through percepts, into the belief base of those agents that are observing that artifact; this is the meaning of: *(i)* the connection between the observable properties and beliefs, and *(ii)* the connection between artifact observable events and **Jason** events. This makes it possible to easily write down **Jason** plans that react to changes in the observable state of an artifact or that are selected based on contextual conditions that include the observable state of possibly multiple artifacts.

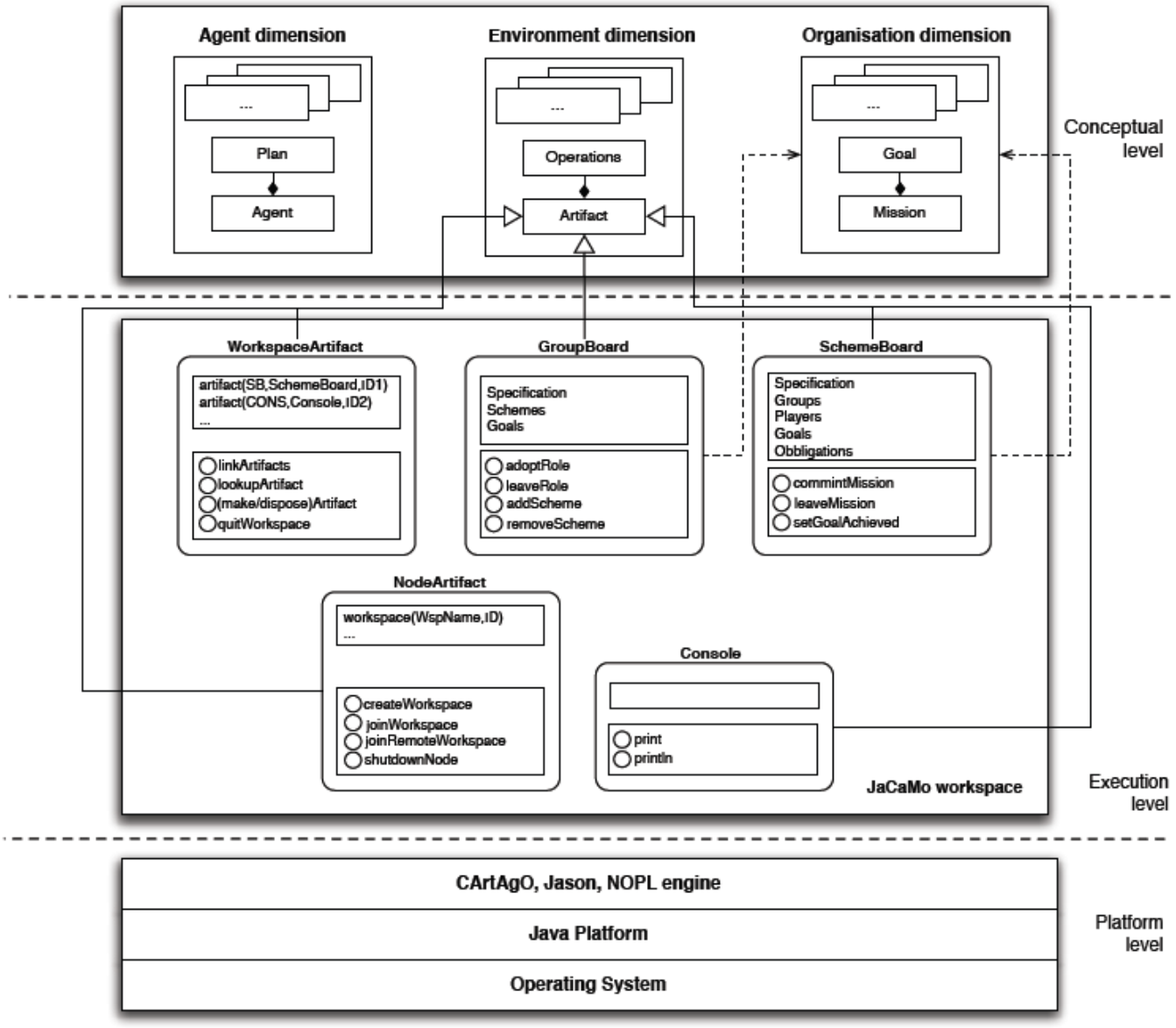
Then, in figure the connections that terminate with a non-filled square represent a set of predefined actions that agents can perform, actions which are mapped into a set of predefined artifacts available in each **JaCaMo** application. These actions refer to the basic functionalities provided by the overall infrastructure, including the environment and organisation layer. This makes it possible in particular to avoid the introduction of *ad hoc*specific mechanisms to exploit infrastructure services concerning the organisation and coordination layer, for instance to adopt a role or to interact with tuple space. Furthermore, since artifacts can be created and disposed dynamically, this makes it possible (also for agents) to update and adapt the infrastructure itself at runtime.

The synergy between the organisation and the environment dimensions (O-E) is based on an organisation management infrastructure for the *Moise* framework [4]. The basic idea is to

uniformly design the organisational infrastructure as part of the (artifact-based) environment in which agents are situated. In such an approach, the different concrete computational entities aimed at managing, outside the agents, the current state of the organisation in terms of groups, social schemes, and normative state are reified in the organisation infrastructure by means of environment artifacts. These artifacts provide a set of basic organisational operations, for example to adopt and leave particular roles, to commit to missions, to signal to the organisation that some social goal has been achieved, etc., through which agents can proactively take part in the organisation. These are shown as dashed connections terminating with a not filled square in figure.

Because of the O-E integration based on the **A&A** meta-model, integration of Agent and Organisation dimensions grounds on the same semantics that maps agents' actions into artifact's operations (e.g., **commitMission**, **adoptRole** and **leaveRole** are all actions that can be performed upon organisational artifacts, see the next figure for details). Similarly, the evolution of the organisational states is made known to the agents through their perception mechanism and the events produced by the various artifacts (e.g., **+player_agents(N) [artifact_name(_,"Group1")]** is an event in which variable **N** is unified with the respective observable property updated by an organisational artifact).

Finally, our approach proposes a mapping from organisational goals into agent goals: this is the meaning of the connection between these two entities shown in the meta-model picture. The state of organisational goals (which agents should fulfill them and when) is computed by an organisational artifact and is displayed as obligations for the agents (e.g., agent **a** is obliged to fulfill the goal **x** in one week). On the agent hand, when such obligations are perceived, and the agent decides to obey, a corresponding local (individual) goal is created.



(Click the image for a larger version)

Summing up, our integrated meta-model promotes a Multi-Agent Oriented Programming approach, made concrete by the **JaCaMo** platform. In this approach, an application is realised by means of a set of **Jason** agents encapsulating the logic and the control of the specific tasks involved in the application, and operating with respect to the organisational constraints defined through appropriate organisational artifacts, and using a set of artifacts which provide the functionalities, and the operations giving access to these functionalities, that agents can employ to do their tasks in the specific context of that application.