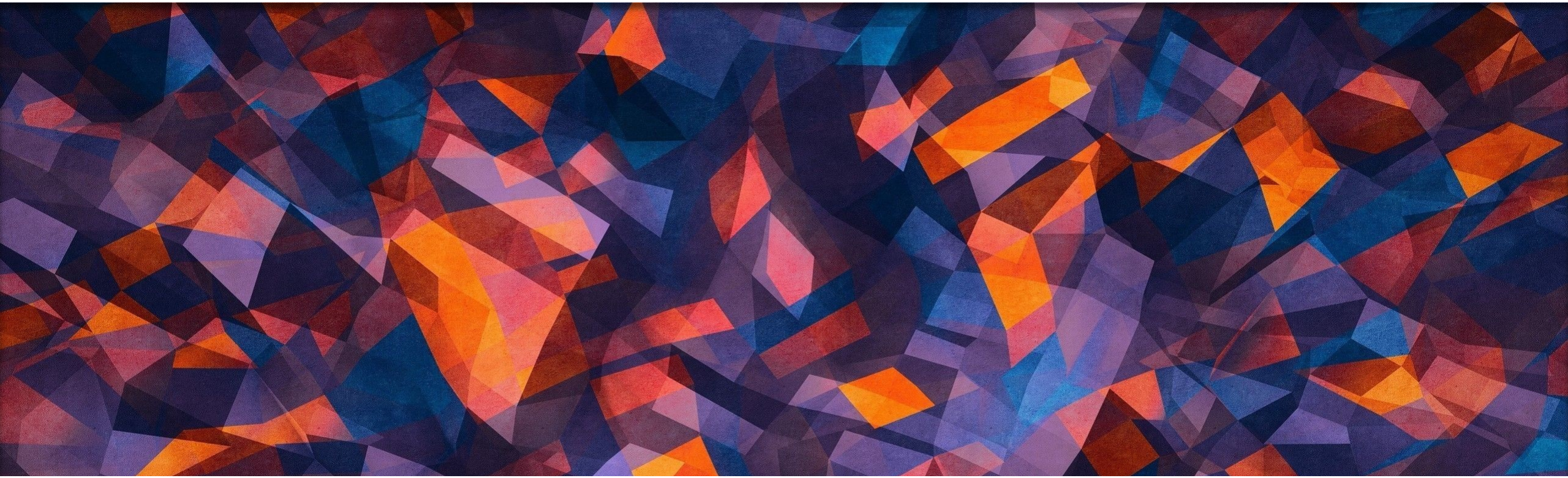
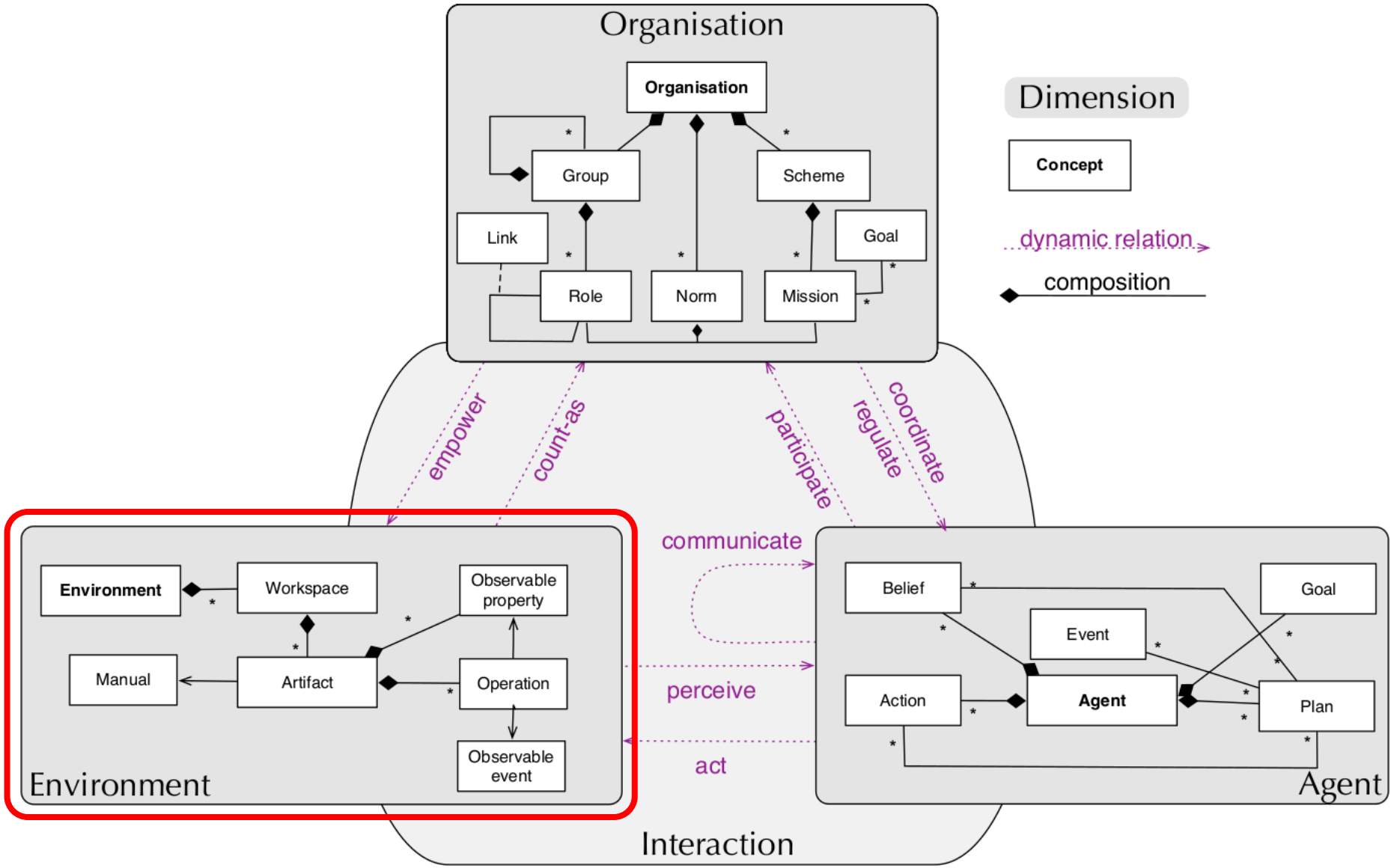


The Environment Dimension

AI4Industry 2023, Saint-Étienne

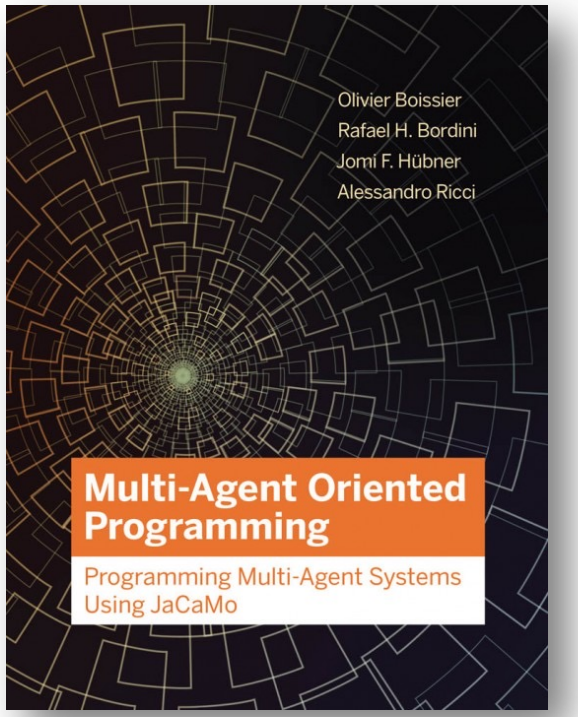
JaCaMo Metamodel – Multi-Agent Concepts



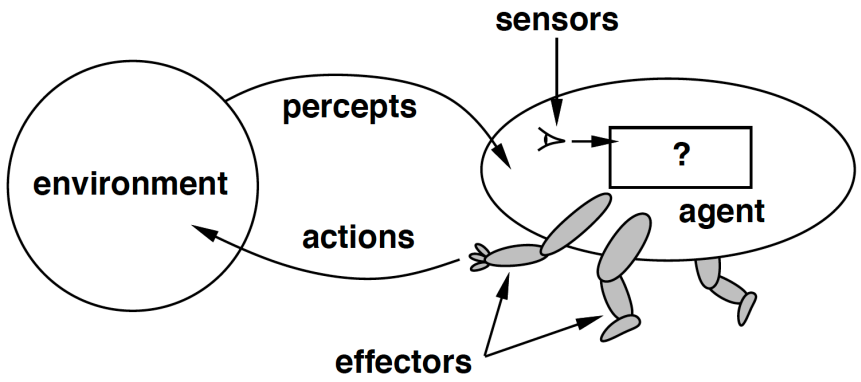
This Session's Agenda

- Environment as a First-Class Abstraction
- The Agents & Artifacts Metamodel

https://www.w3.org/community/webagents/

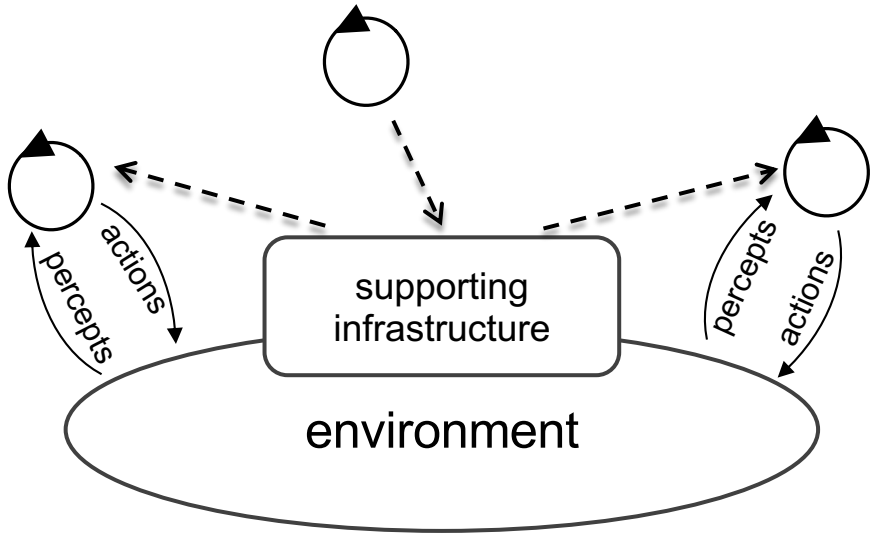


The Environment Dimension



Single-agent system perspective
[Russell & Norvig, 2020]

The **Environment** as the **world external to the system**



Multi-agent system perspective

The **Environment** becomes **part of the system**
(e.g.: communication and interaction infra.)

Stuart Russell and Peter Norvig (2020) Artificial Intelligence: A Modern Approach (Fourth Edition).
D. Weyns, A. Omicini, and J. Odell. Environment as a first class abstraction in multiagent systems. JAAMAS 14, 5–30, 2007.

The Environment as a Design Abstraction

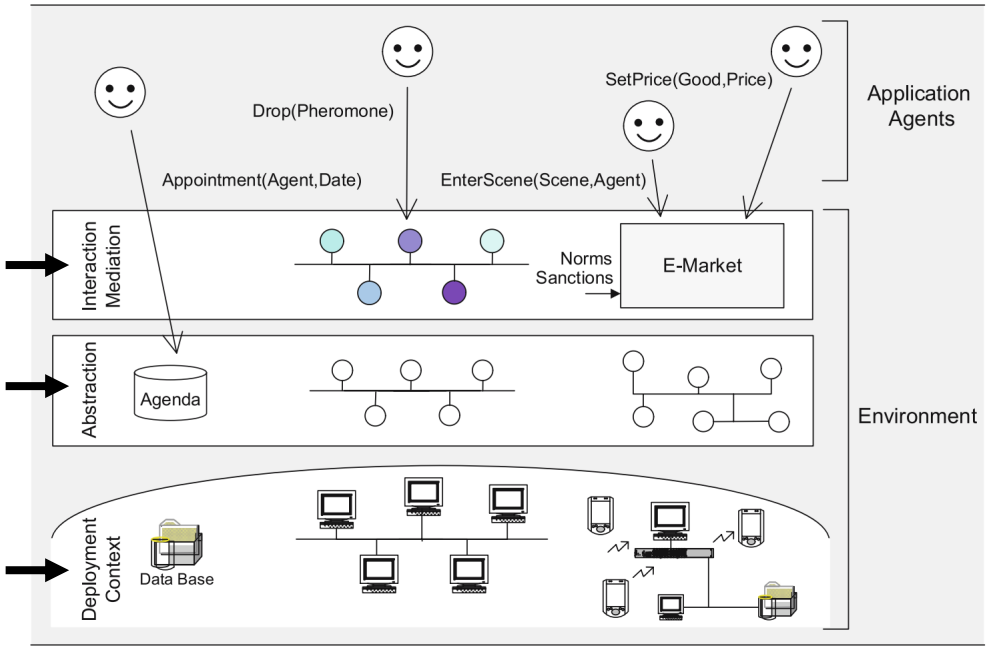
The **environment is a first-class abstraction** that provides the surrounding conditions for agents to exist and that mediates both the interaction among agents and the access to resources [Weyns et al., 2007].

3 levels of support:

Interaction-mediation →

Abstraction →

Basic →



Engineering MAS: environment as a **first-class design abstraction** [Weyns et al., 2007].

Reflection support [Rici et al., 2011]: mechanisms to modify the functional behavior of the environment

- Example: creating and destroying artifacts

Interaction-mediation support: mechanisms to mediate, enact, and regulate interactions

- Example: pheromone infrastructure, e-institutions, rate limiting, etc.

Abstraction support: conceptual bridge between abstractions used to design and program agents and the deployment context

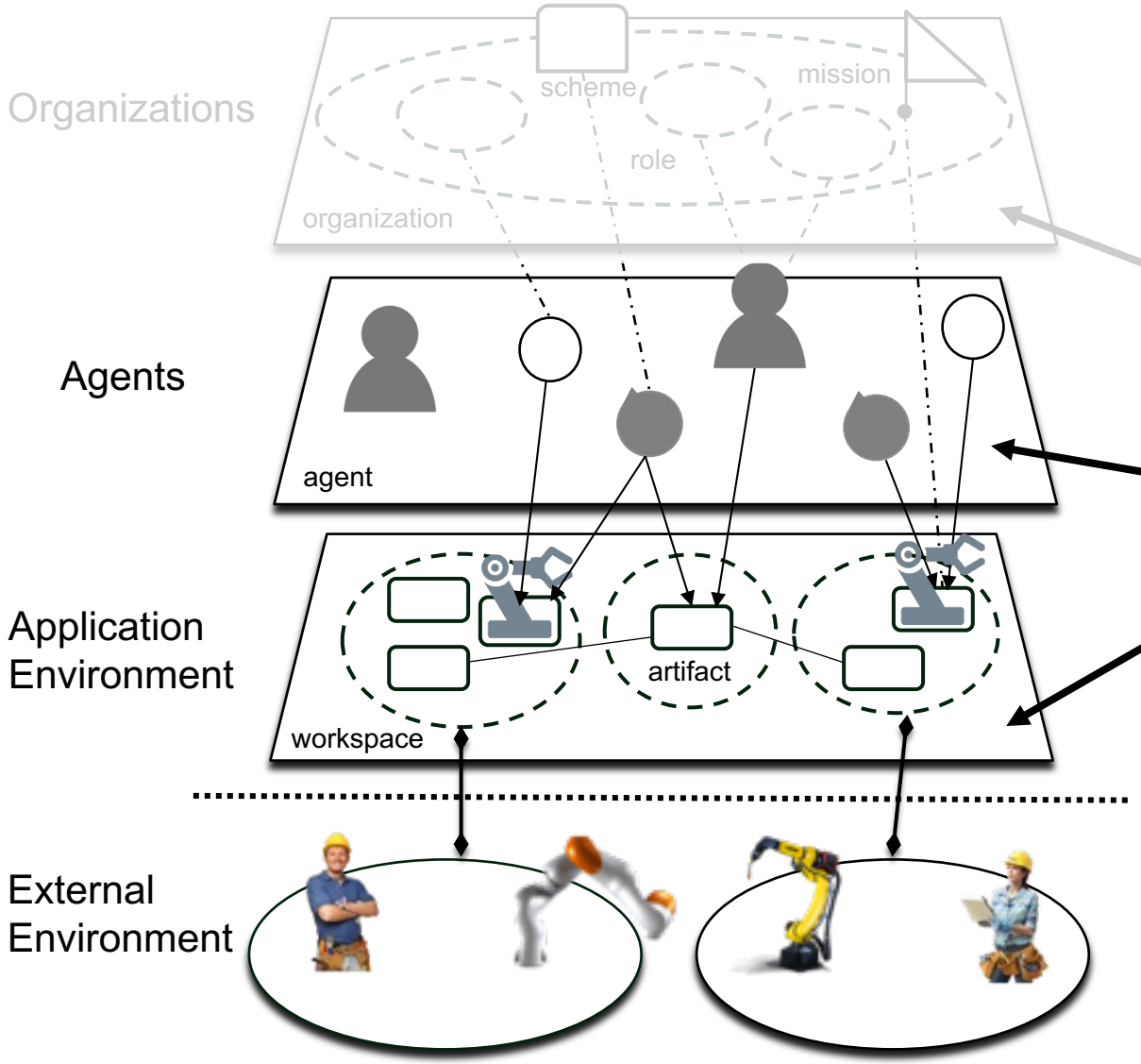
- Example: semantic models, domain-specific abstractions, etc.

Basic interface support: raw access to the deployment context

- Example: Web APIs, device interfaces, etc.

D. Weyns, A. Omicini, and J. Odell. Environment as a first class abstraction in multiagent systems. JAAMAS 14, 5–30, 2007.
A. Ricci, M. Piunti, and M. Viroli. Environment programming in multi-agent systems: an artifact-based perspective. JAAMAS 23, 158–192, 2011.

Example: Flexible Industrial Manufacturing



SIEMENS

collectives of people and artificial agents working towards shared goals

autonomous behavior

non-autonomous behavior

Modularity and **Encapsulation**

Uniformly **encapsulate** and **modularize** functionalities of the MAS outside of the agents

⇒ **reusability** of components

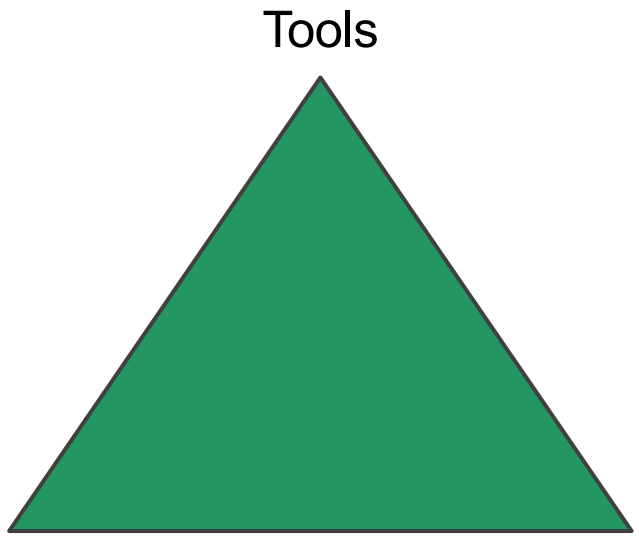
⇒ system **evolvability**: components can be developed, deployed, and can evolve independently

Andrei Ciortea, Simon Mayer, and Florian Michahelles. Repurposing Manufacturing Lines on the Fly with Multi-Agent Systems for the Web of Things, AAMAS 2018.

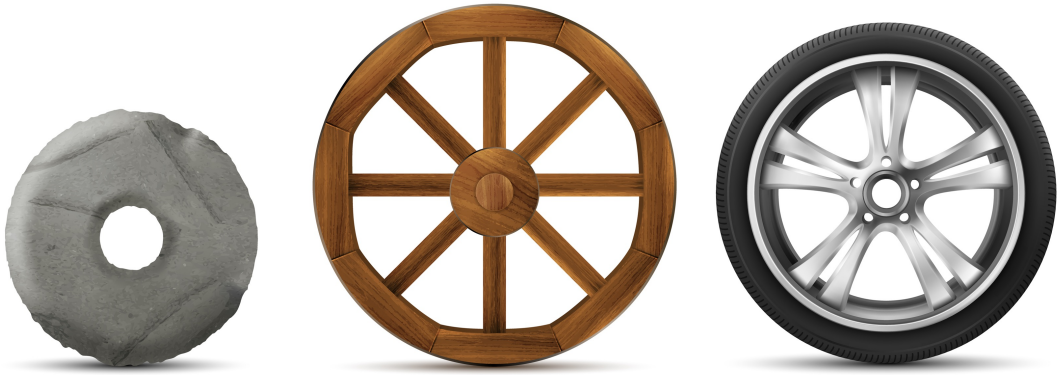
This Session's Agenda

- Environment as a First-Class Abstraction
- The Agents & Artifacts Metamodel

Activity Theory



Roots in cultural-historical psychology (1920s and 1930s)

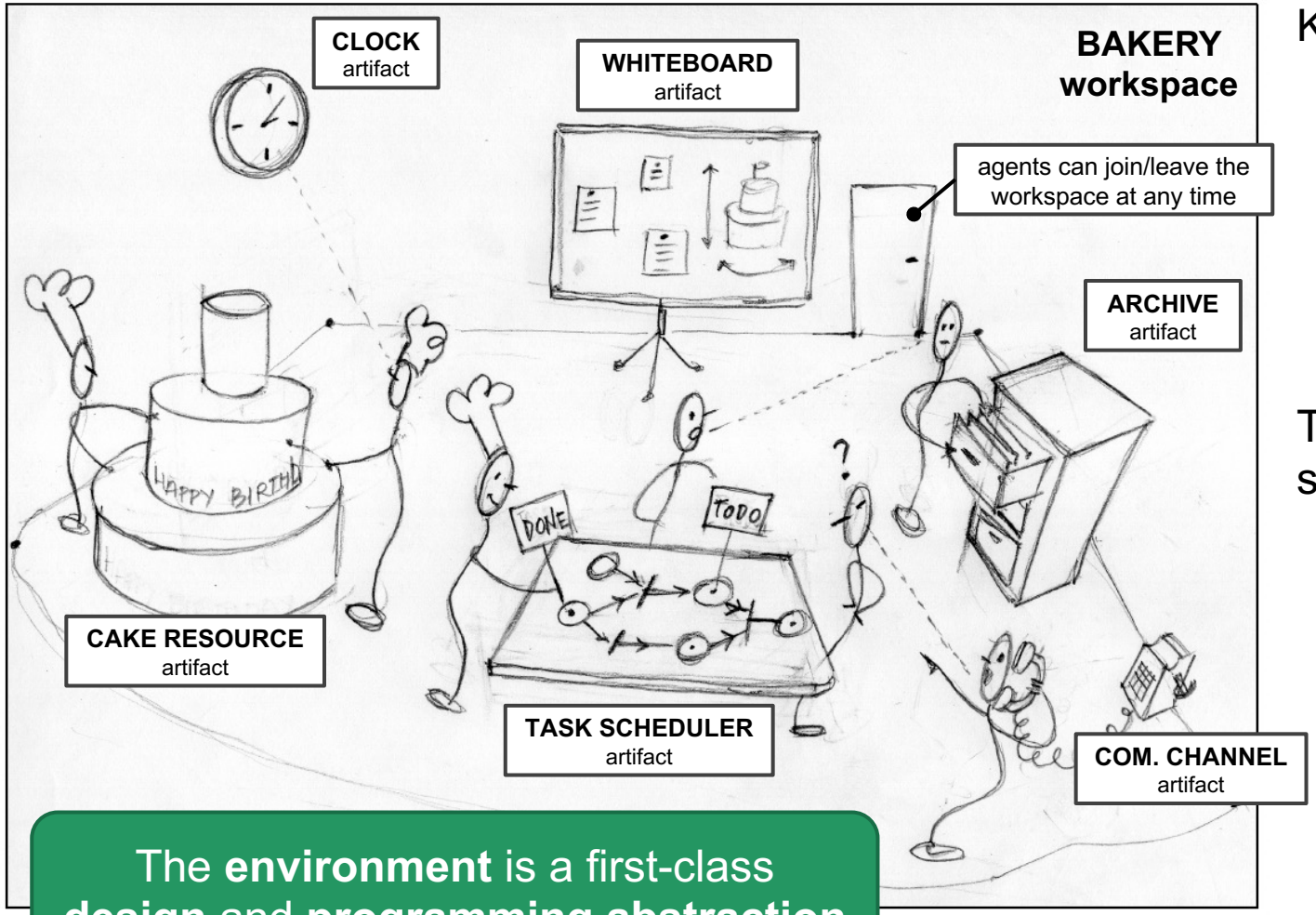


Activity (basic unit of analysis) is a **goal-directed interaction** with the world

The activity is mediated through **tools** (or **artifacts**), which evolve over time based on the experience of subjects

Bonni A. Nardi. *Context and Consciousness: Activity Theory and Human-Computer Interaction*. MIT Press, 1996.

The Agents & Artifacts Metamodel



The **environment** is a first-class **design** and **programming abstraction**

Key idea: **separation of concerns**

- **agents** encapsulate **autonomous** behavior
- **artifacts** encapsulate **non-autonomous** behavior

Programming MAS = Programming **Agents**
+ Programming the **Environment**

The agents' environment is modelled as a **dynamic** set of **artifacts** grouped into **workspaces**

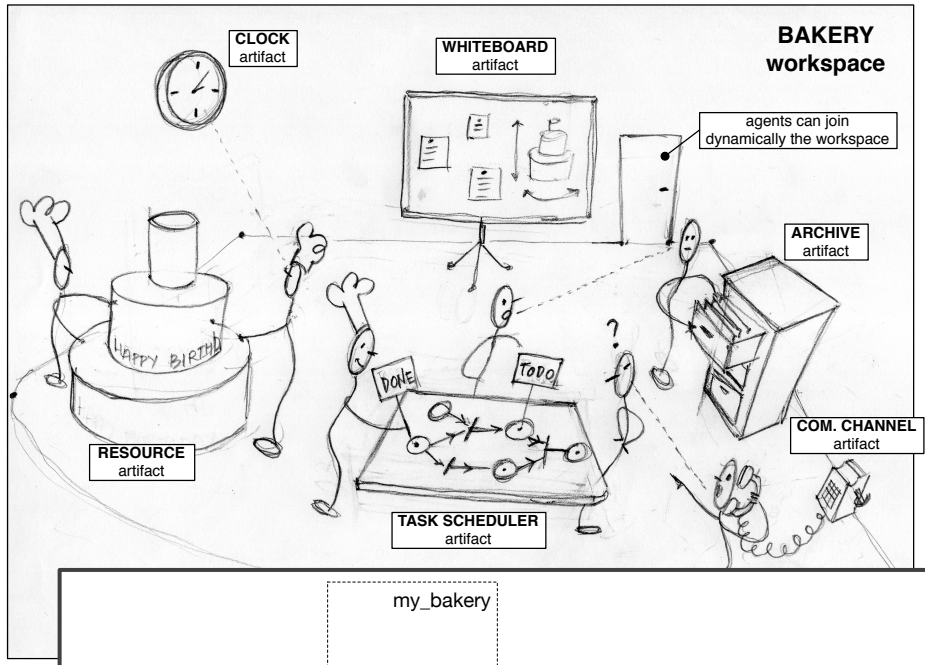
- the **actions** provided to agents are determined by the artifacts **discovered at run time**
- agents **construct**, **share**, and **use** artifacts to support their working activities

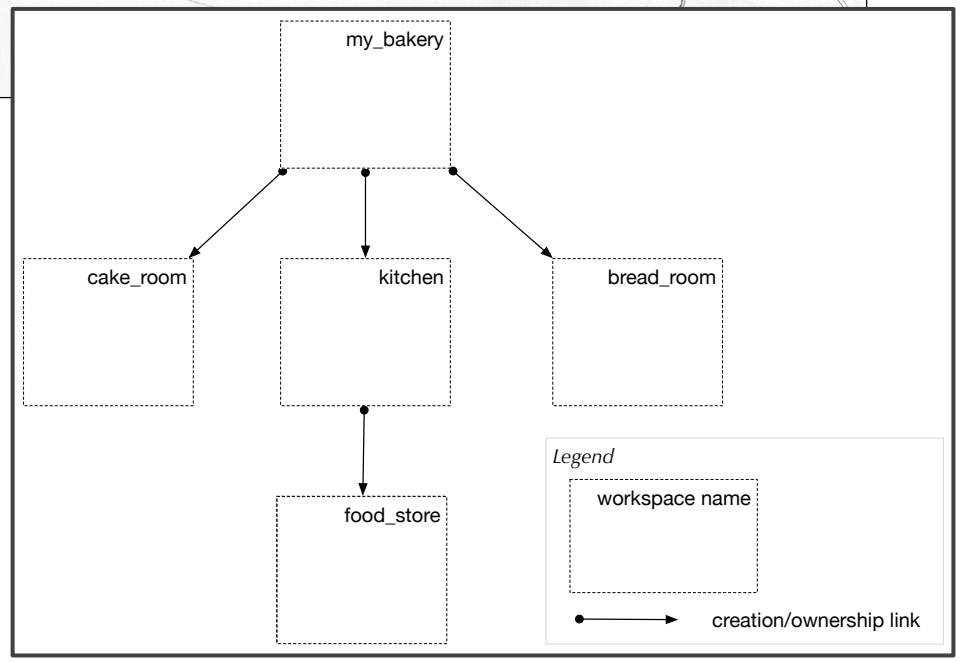
⇒ artifacts are **mediating tools** for goal-directed agents

⇒ agents can **modify** the **functional behavior** of the environment to meet their needs

O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

The Workspace Abstraction





A **logical place** containing artifacts and the working context of the agents' activities

- provides a notion of **locality** and **situatedness**
- allow to **structure** complex/distributed environments

Agents can **join**, **leave**, and **work in** multiple workspaces at the same time

- agents are **embodied** and interact within the workspace through **body artifacts**

⇒ separation of concerns between the **agent's mind** and the **agent's body**

⇒ allows **heterogeneous agents** (implementing different architectures) to *join* and *work in* the same environment

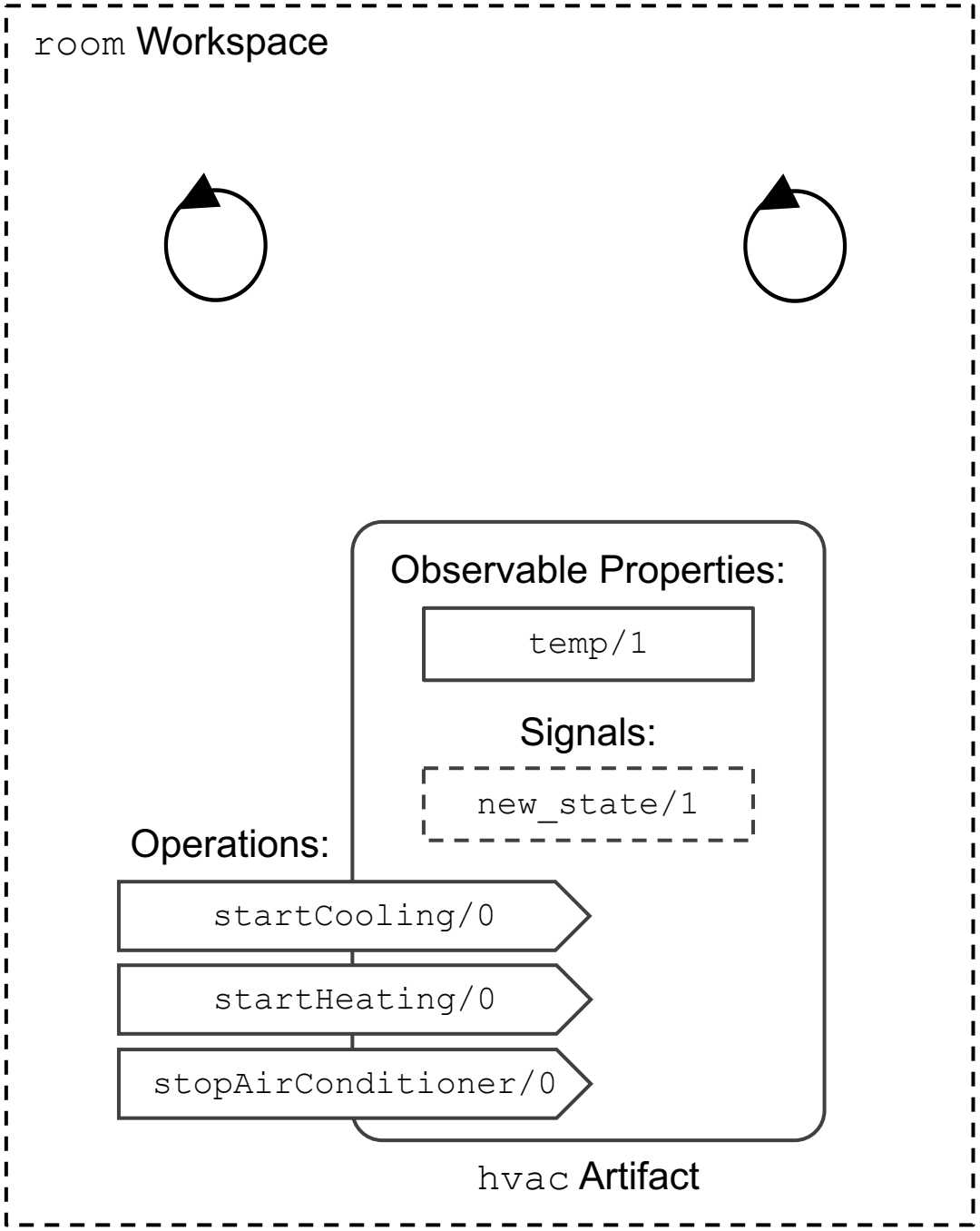
Workspaces can be distributed over a network

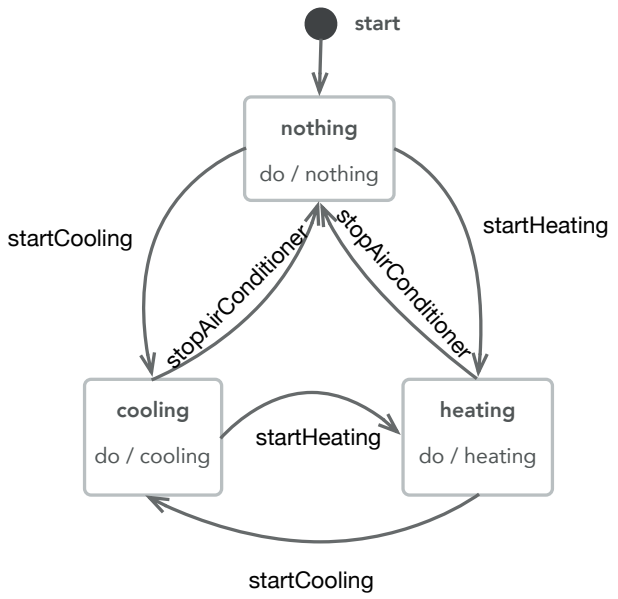
Alessandro Ricci, *Levels of Abstraction in Designing and Programming Systems of Cognitive Agents*, HyperAgents 2019: http://www2019.hyperagents.org/

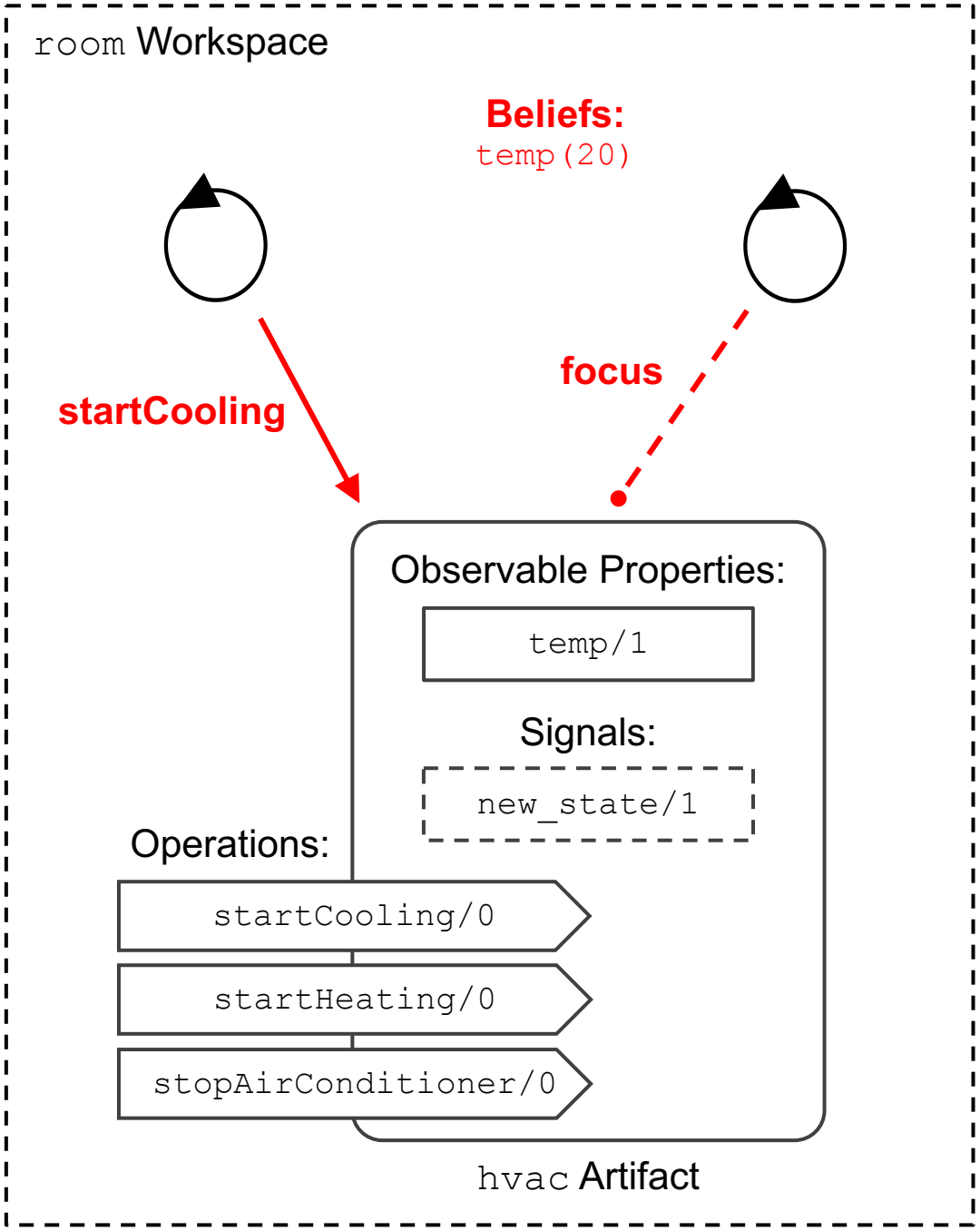
The Artifact Abstraction

Artifacts as computational objects

- *usage interface*:
  - **observable properties**: state variables that can be perceived by agents
  - **observable events**: non-persistent signals that carry information and can be perceived by agents
  - **operations**: environmental actions provided to the agent
    - operations can update the values of observable properties or can generate signals





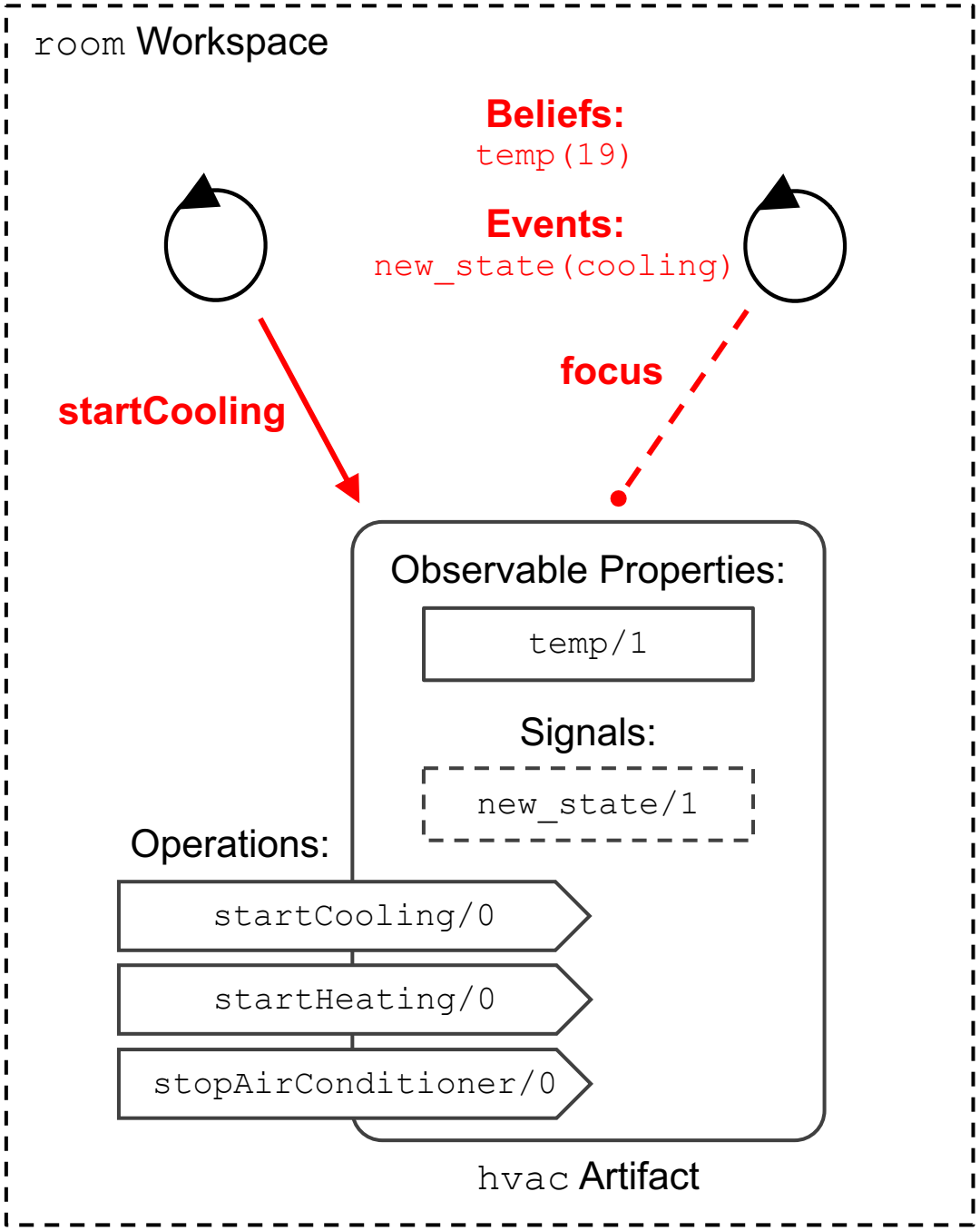


The Artifact Abstraction

Artifacts as computational objects

- *usage interface*:
  - **observable properties**: state variables that can be perceived by agents
  - **observable events**: non-persistent signals that carry information and can be perceived by agents
  - **operations**: environmental actions provided to the agent
    - operations can update the values of observable properties or can generate signals

Agents can **focus** on artifacts to **perceive** observable properties and signals



The Artifact Abstraction

Artifacts as computational objects

- *usage interface*:
  - **observable properties**: state variables that can be perceived by agents
  - **observable events**: non-persistent signals that carry information and can be perceived by agents
  - **operations**: environmental actions provided to the agent
    - operations can update the values of observable properties or can generate signals

Agents can **focus** on artifacts to **perceive** observable properties and signals

The Artifact Abstraction

Artifacts as computational objects

- *usage interface*:
  - **observable properties**: state variables that can be perceived by agents
  - **observable events**: non-persistent signals that carry information and can be perceived by agents
  - **operations**: environmental actions provided to the agent
    - operations can update the values of observable properties or can generate signals

Agents can **focus** on artifacts to **perceive** observable properties and signals

Why is **intentional focus** useful?

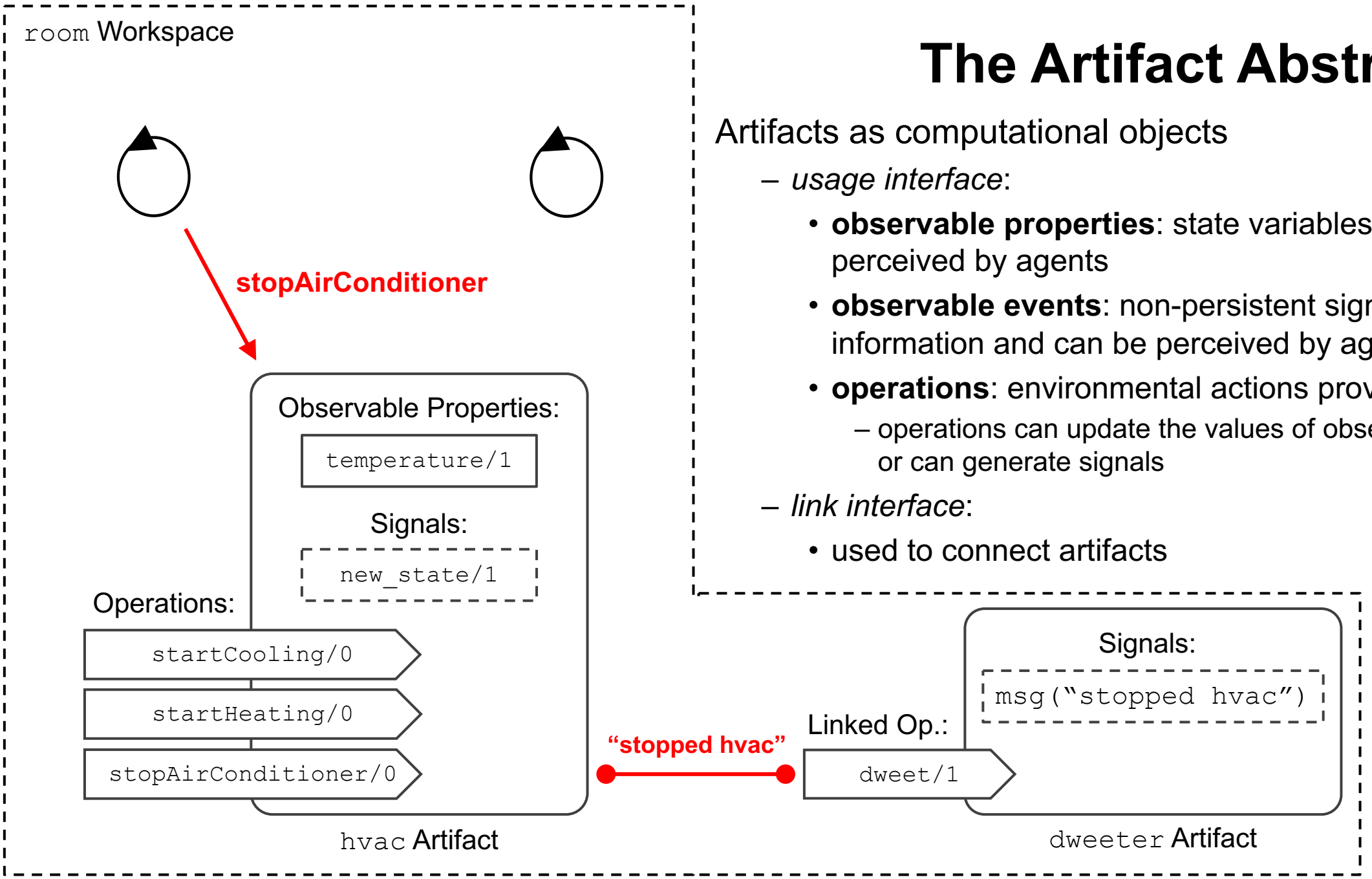
Allows agents to **select** the parts of the environment that are relevant to their goals

- promotes **scalability**
  - agents can cope with larger environments
  - the environment infrastructure can serve more agents
- promotes **autonomy** from the environment

The Artifact Abstraction

Artifacts as computational objects

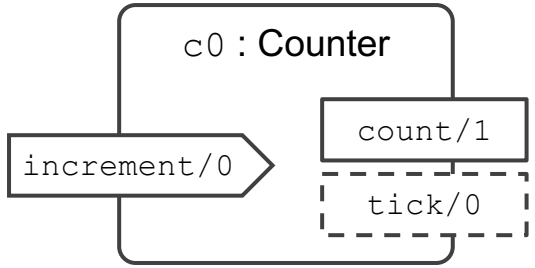
- *usage interface*:
  - **observable properties**: state variables that can be perceived by agents
  - **observable events**: non-persistent signals that carry information and can be perceived by agents
  - **operations**: environmental actions provided to the agent
    - operations can update the values of observable properties or can generate signals
- *link interface*:
  - used to connect artifacts



A Basic Taxonomy of Artifacts

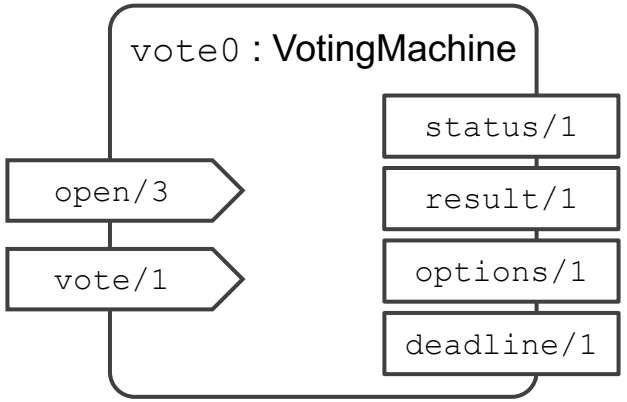
Resource Artifacts

– some specific kind of resource that can be shared by agents



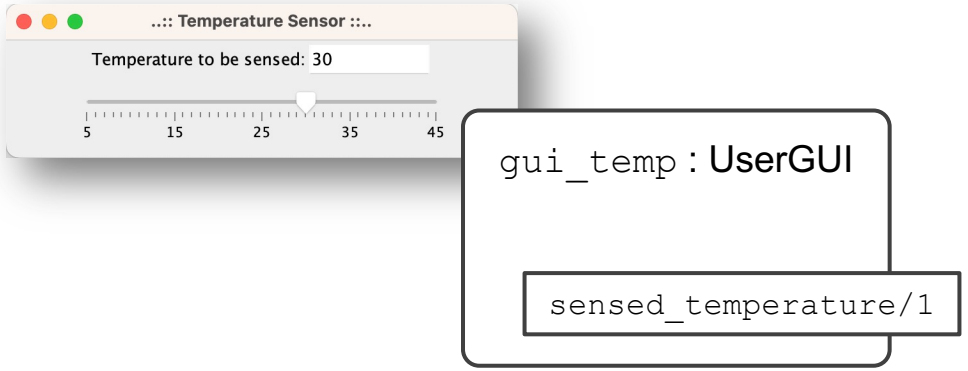
Coordination Artifacts

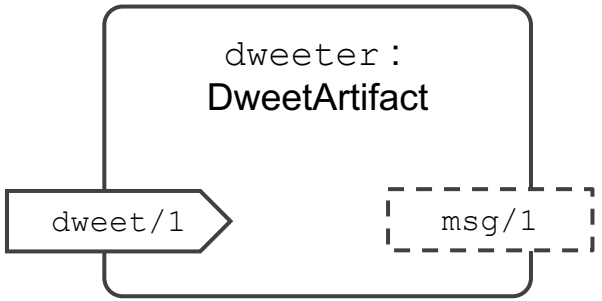
– artifacts specifically designed to provide coordination functionalities by enabling and managing in some way the interaction among agents

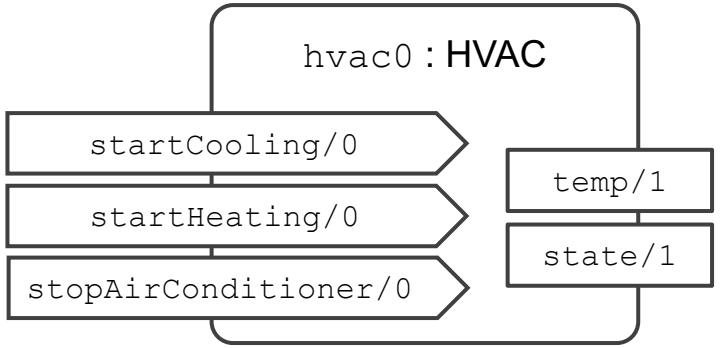


Boundary Artifacts

– artifacts that allow agents to interact with human users and, more generally, any actor or system that is external with respect to the MAS

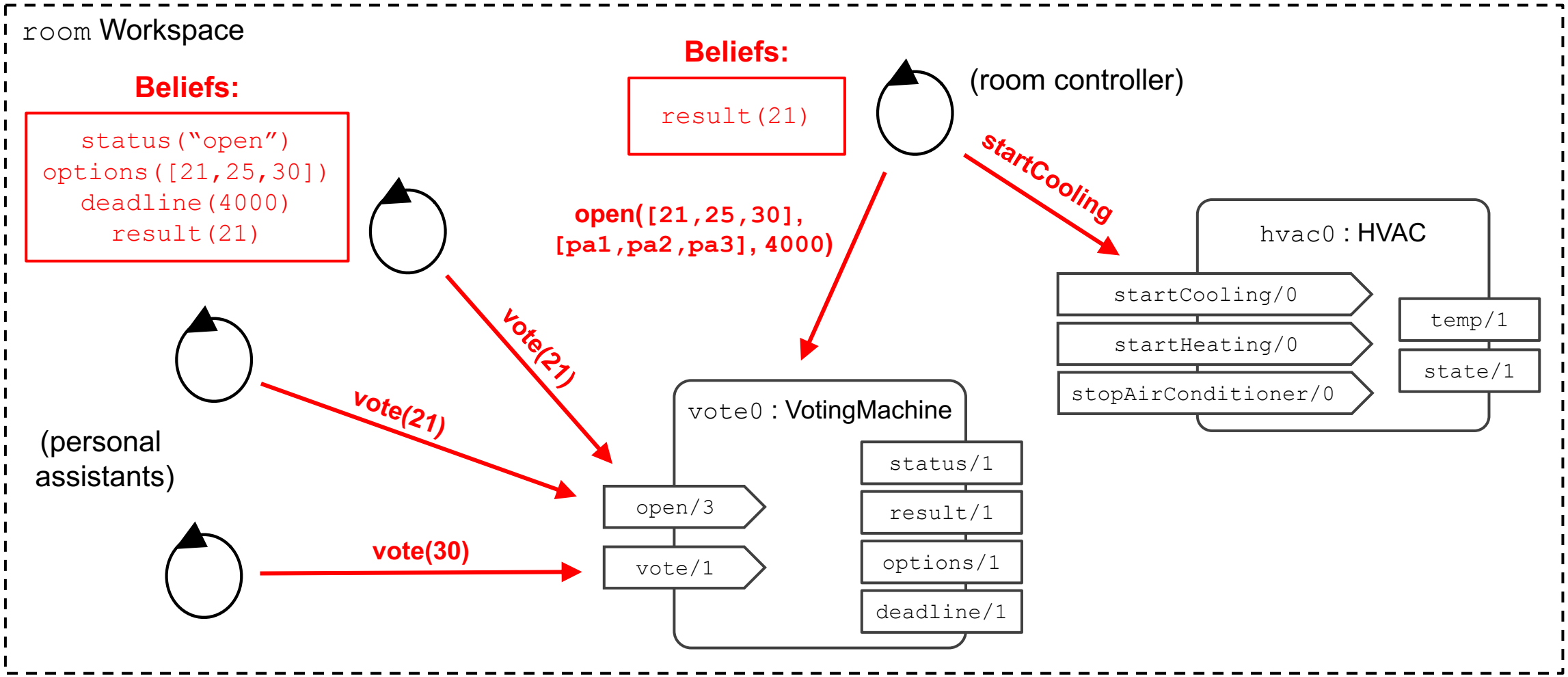






O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

Smart Room Scenario Revisited: Voting Machines



O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

Artifacts vs. Objects

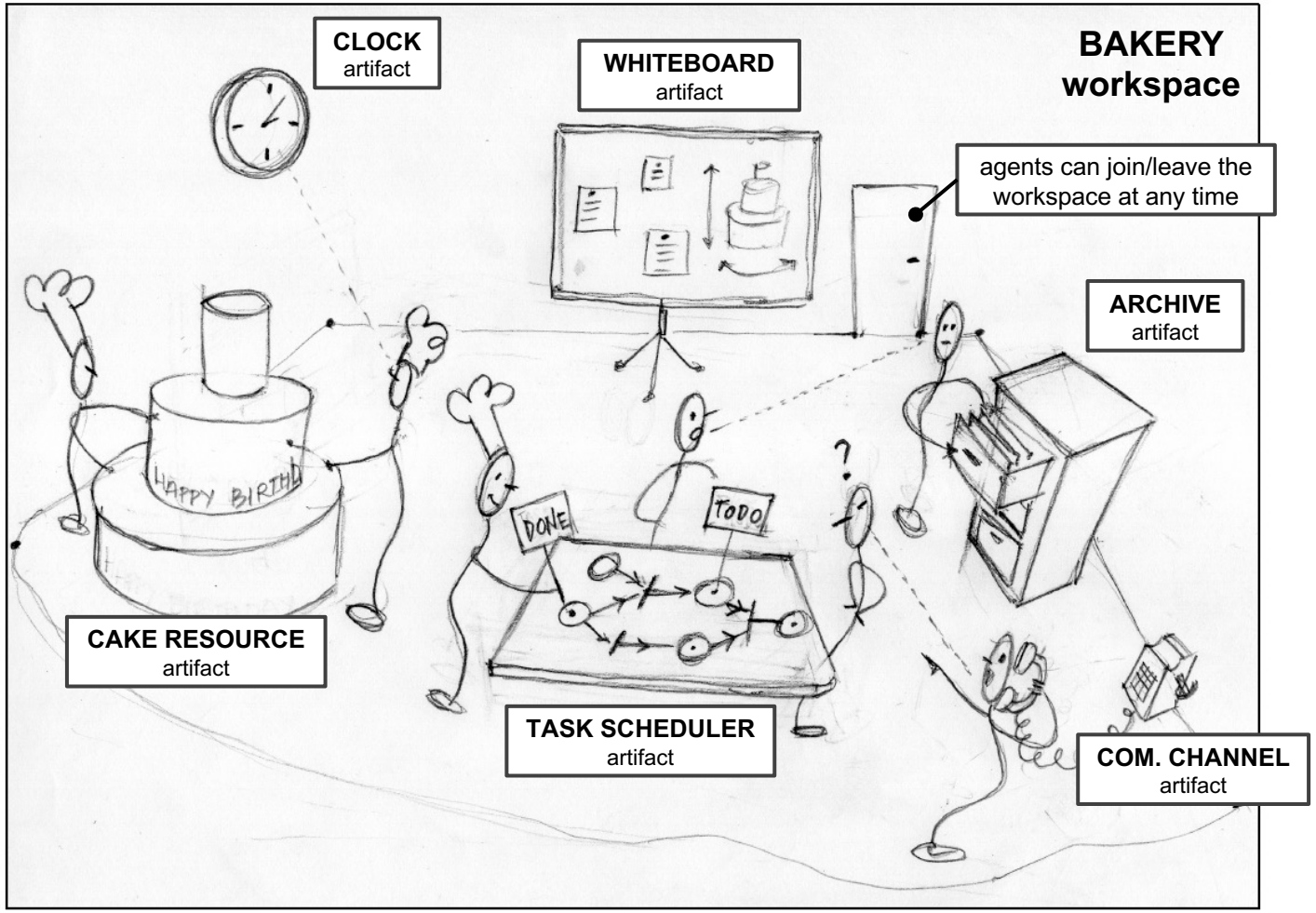
Both artifacts and objects model **nonautonomous entities** and provide a **usage interface**

But there are important differences:

- **transfer of control**:
  - in object-object interaction, a method call **implies a transfer of control** between the caller object and the callee object
  - in agent-artifact interaction, **control is encapsulated inside agents** and cannot be transferred
    - the execution of a triggered operation is carried out by another logical flow provided by the environment
    - on the agent side, the plan in execution is suspended until the action is either completed or failed (the agent can continue to pursue other intentions)
- **observable state:**
  - artifacts **have observable state** captured by observable properties
  - unlike public object instance fields, observable properties cannot be written directly (they can be updated by operations)
- **concurrency:** artifacts are **thread-safe by design**, which makes it easy to share them among agents

O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

The Agents & Artifacts Metamodel

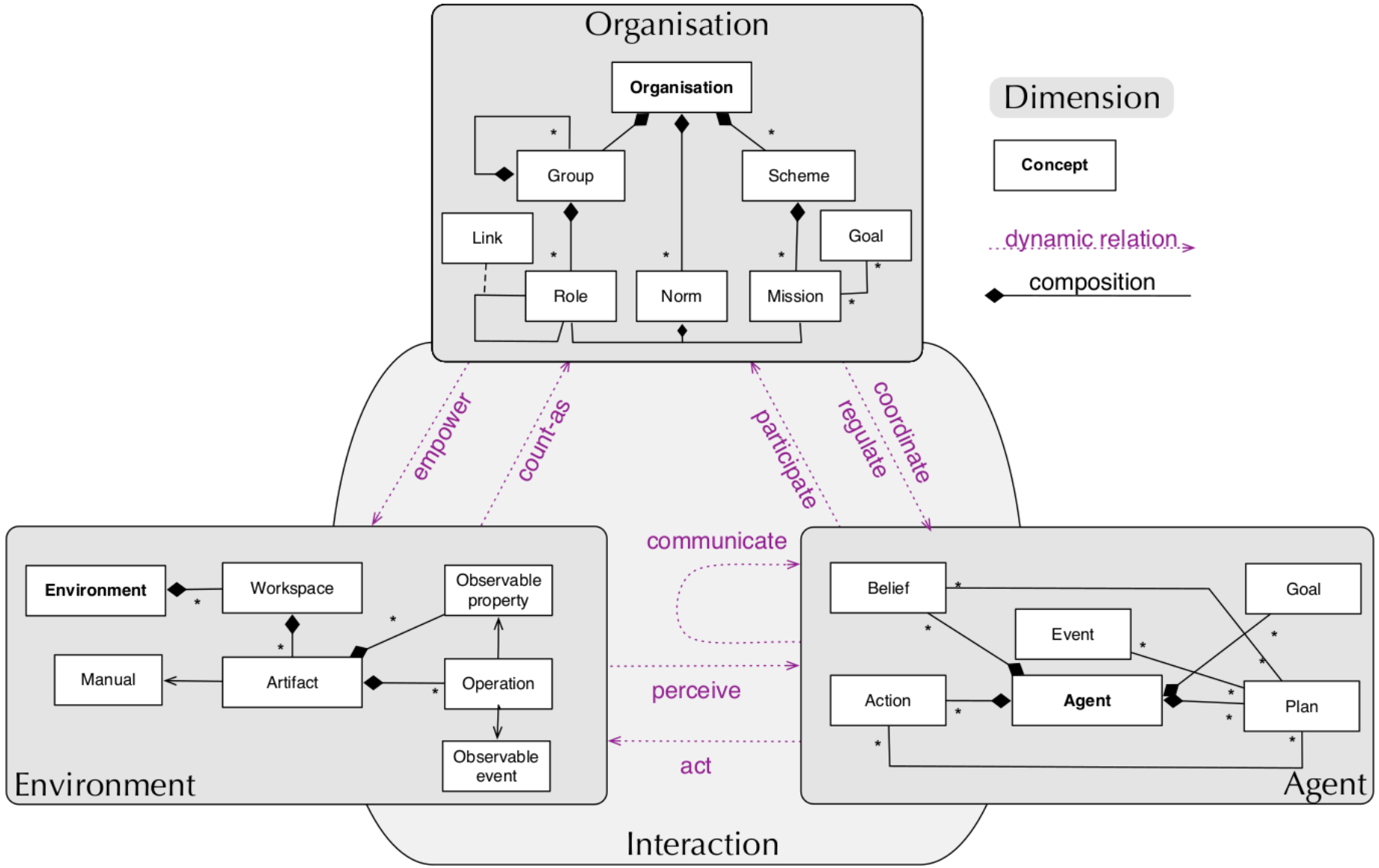


The **environment** is a first-class **design** and **programming abstraction**

Programming MAS = Programming **Agents** + Programming the **Environment**

O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

JaCaMo Metamodel – Multi-Agent Concepts



Any Questions / Comments / Doubts / Concerns?

Images

https://freepik.com