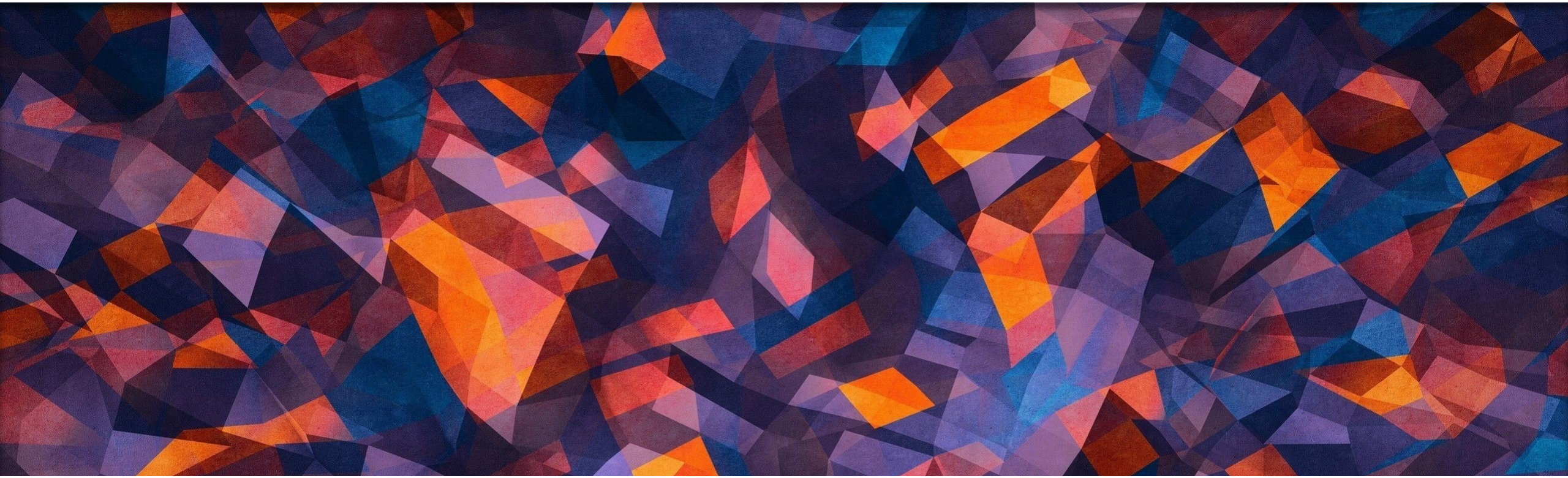
Universität St.Gallen

Reflecting the best of human society & evolution (ants) onto digital systems ⟶ rational systems



Web-based Autonomous Systems

Coordination III: Normative Multi-Agent Systems and Organizations

Chair for Interaction- and Communication-based Systems (ICS-HSG)

Universität St.Gallen

Our Journey

Prerequisites:
- **ASSE**
- **(…)**

Week 1:
Introduction

Week 2:
A Web for Machines

Week 3:
Knowledge Representation and Reasoning for the Web

Week 4:
Linked Data and Distributed Knowledge Graphs

Week 5:
Autonomous Agents (Arch. and Programming)

Week 6 (Coordination I):
Agent Communication and Interaction

Week 7 (Coordination II):
Self-Organization and Stigmergy

Week 8 (Coordination III):
Normative MAS and Organizations

Week 9 (Coordination IV):
Trust & Reputation

Week 10:
Game Theory and Social Choice

Week 11:
Reinforcement Learning and Multi-Agent Learning

Week 12:
An Industry Perspective

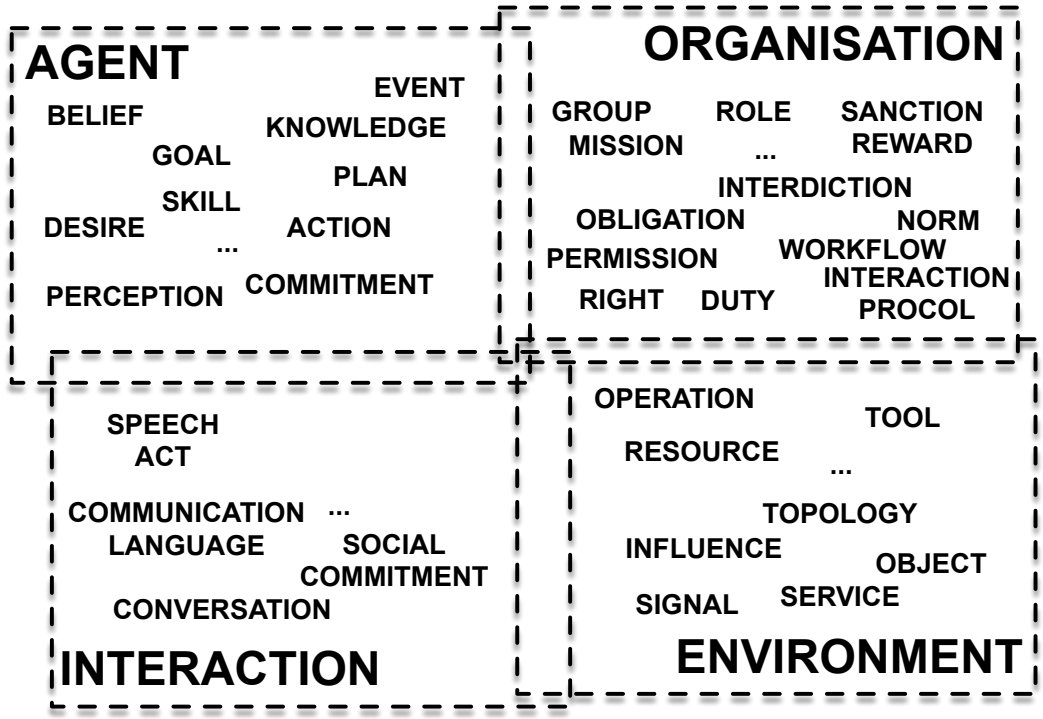
Exercises

Ex1: Writing Your First Agent(s)!
Ex2: Automated Planning
Ex3: Web Ontologies
Ex4: Operating on Linked Data
Ex5: BDI Agents
Ex6: Interacting Agents on the Web
Ex7: Ant Colony Optimization
Ex8: Organized Agents
Ex9: Trustworthy Agents
Ex10: Axelrod's Agents
Ex11: Reinforcement Learning Agents
Course Review and Q&A

Universität St.Gallen

Coordination Perspectives

Week 9 (Coordination IV):
Trust & Reputation and Multi-Agent Planning



Week 8 (Coordination III):
Normative MAS and Organizations

Week 6 (Coordination I):
Communication and Interaction

Week 7 (Coordination II):
Self-Organization and Stigmergy

Modelling Dimensions for Multi-Agent Systems
[Demazeu, 1995]

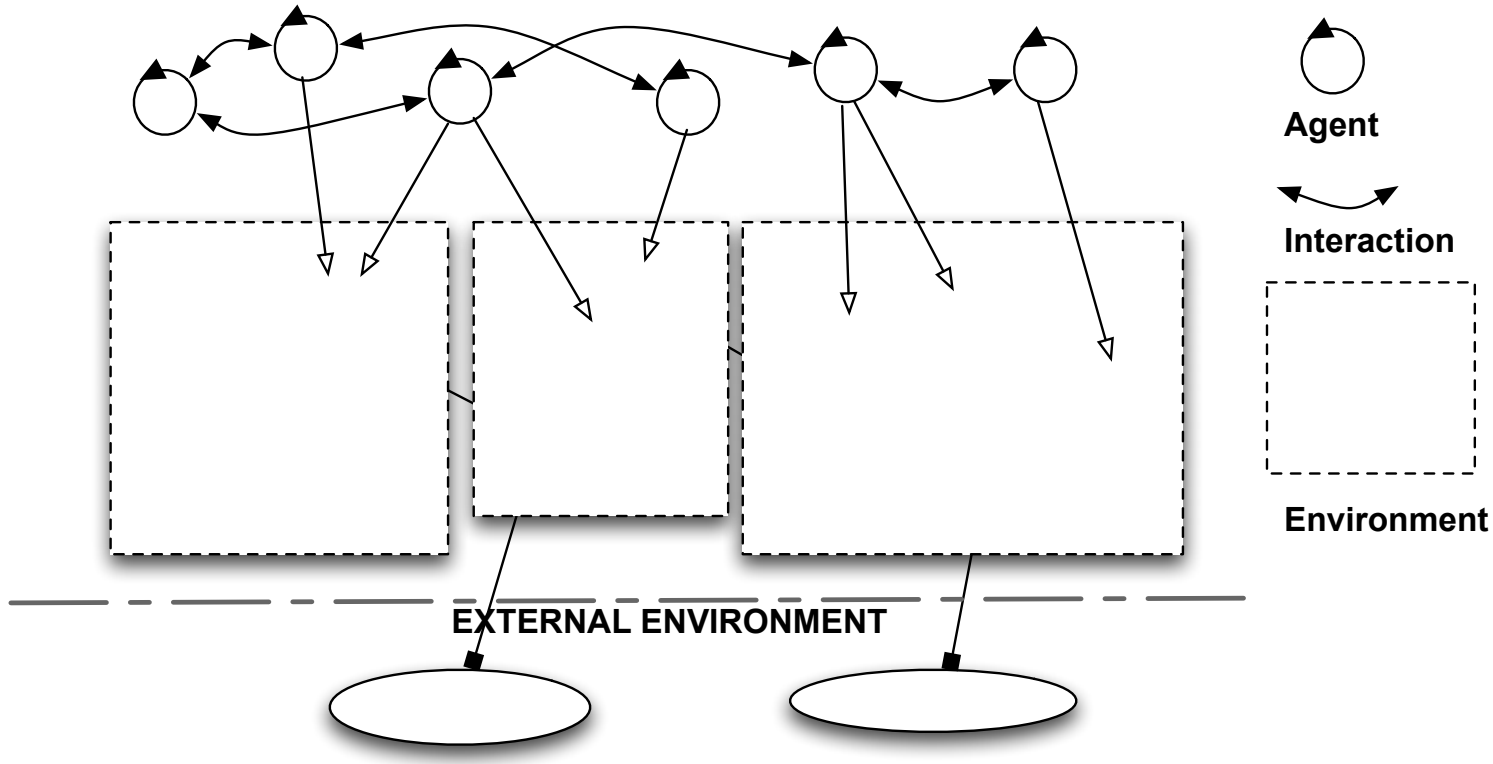
O. Boissier, *Multi-Agent Coordination Course*, MINES Saint-Étienne, 2021.
Y. Demazeau. From Interactions To Collective Behaviour In Agent-Based Systems, 1st European Conf. on Cognitive Science, 1995.

Autonomy & Control: Bottom-up vs. Top-down

Local interactions create **global emergent behavior**

How to enable **control** over the emergent behavior?

How to achieve a desired global behavior **efficiently**?

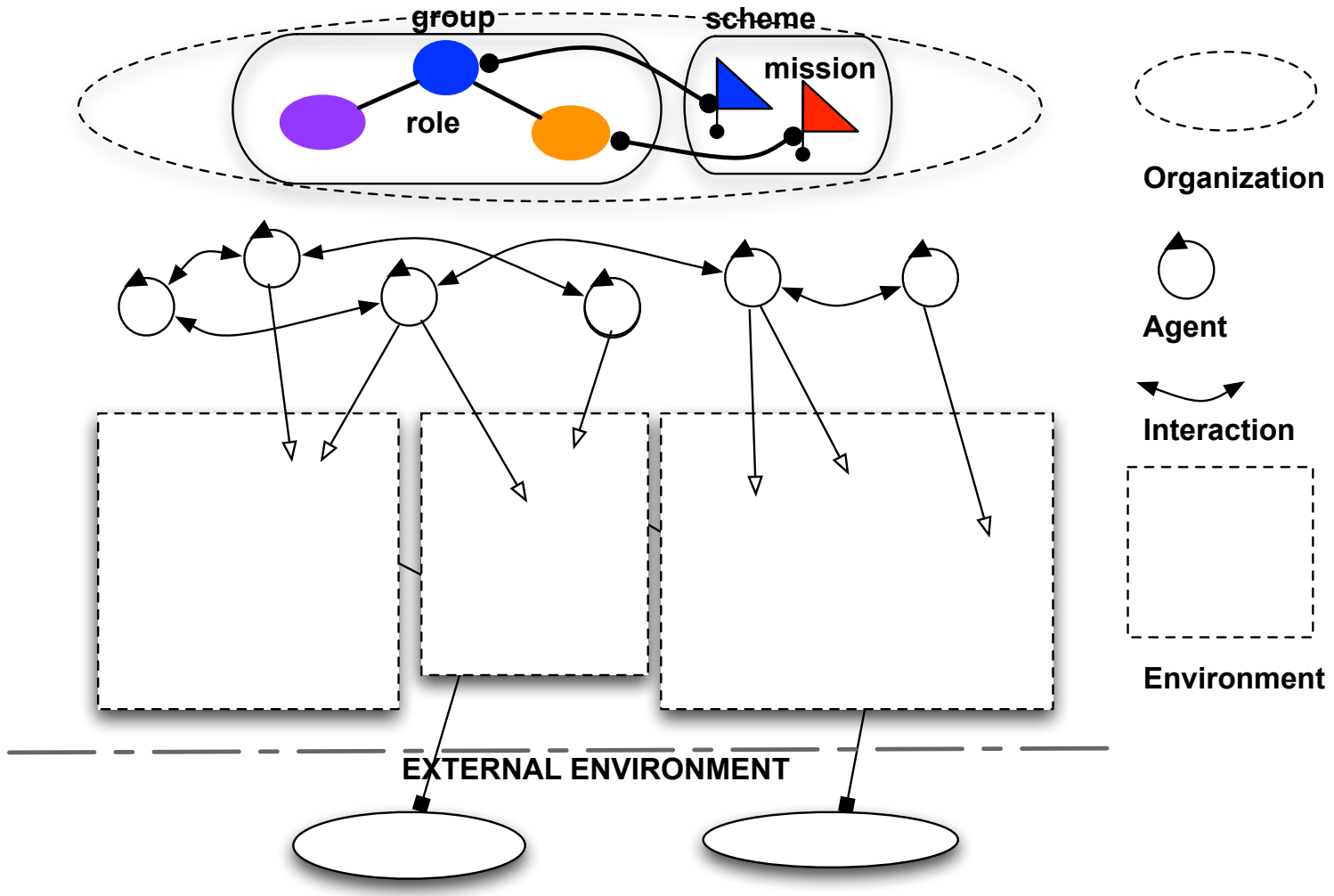


Autonomy & Control: Bottom-up vs. Top-down

Local interactions create **global emergent behavior**

How to enable **control** over the emergent behavior?

How to achieve a desired global behavior **efficiently**?

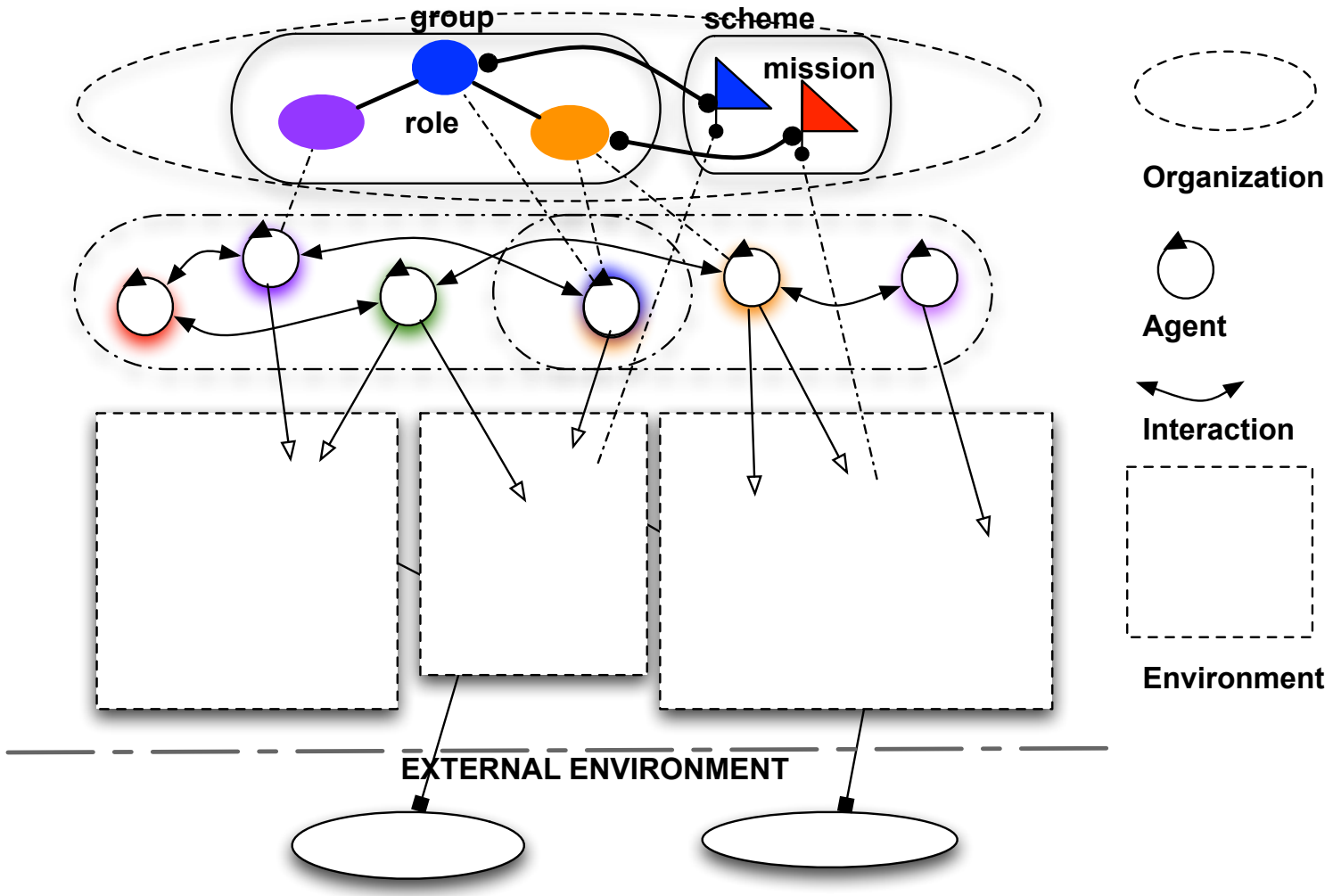


Autonomy & Control: Bottom-up vs. Top-down

Local interactions create **global emergent behavior**

How to enable **control** over the emergent behavior?

How to achieve a desired global behavior **efficiently**?

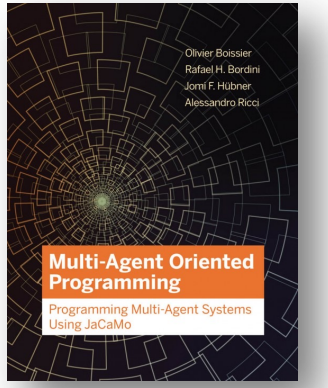


Universität St.Gallen

Today's Agenda

- Normative Multi-Agent Systems
- Normative Organizations
  - Organization-Oriented Programming
  - The Moise Organization Model
  - Hands-on: Programming Organizations with JaCaMo

⇒ revisiting Autonomy from WO1/WO2



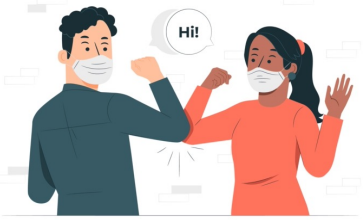
Chapters 8-9

Norms in Human Society

Regulative Norms

Constitutive Norms

raising hand in ⟨ auction → buy / street → taxi



Norms in Everyday Interactions

can change

A baseline definition [Verhagen et al., 2018]

"**Norms** are a **regularity** of **intentional behavior** within a **certain population**."
(i.e., norms provide the framework within which members of a society interact)

Interaction: norms are at the **interface** between the **individual** and **society**

Agency vs. **Structure**: the **intentional behavior** is a **regular pattern** within a group

H. Verhagen, M. Neumann, and M.P. Singh. Normative Multiagent Systems: Foundations and History. Handbook of Normative Multi-Agent Systems. College Publications. 2018.

Universität St.Gallen

Normative Multi-Agent Systems

Norms define a standard of behavior within a society of agents

Norms

Norms are **rules** defined within the scope of a society in order to influence the behavior of human and/or artificial agents.

Normative Multi-Agent System (MAS) [Boella et al., 2008]

A MAS **organized** by means of mechanisms to **represent**, **communicate**, **distribute**, **detect**, **create**, **modify**, and **enforce norms**, and mechanisms to **deliberate about norms** and **detect norm violation and fulfillment**.

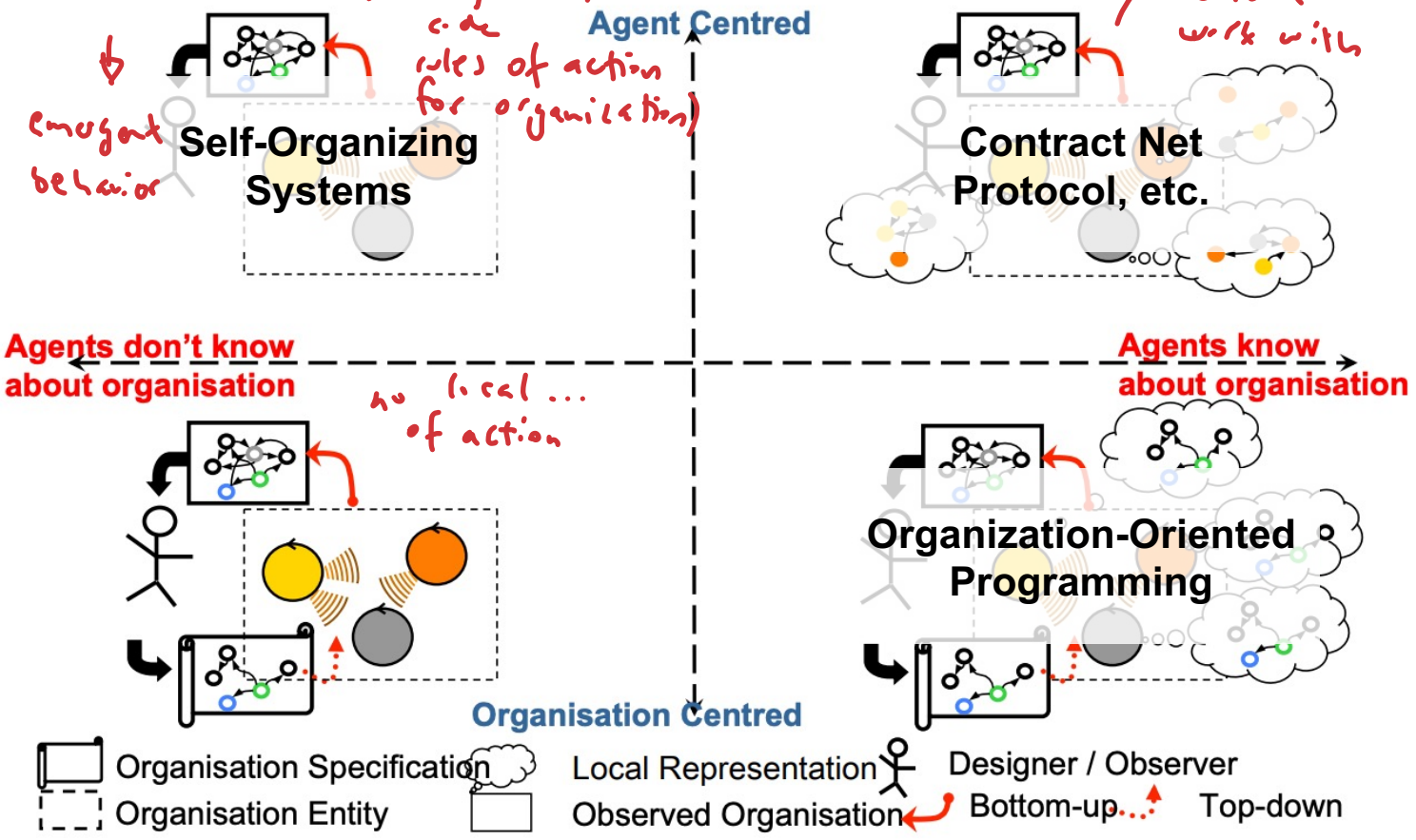
Deductive : Reason from rules (?)

Induction : Can have false positives (!) ~~~ problem w/ chat GPT ~~~ "Norm" is defined by OpenAI

O. Boissier, *Multi-Agent Programming Course*, MINES Saint-Étienne, 2021.
G. Boella, L. Torre, H. and Verhagen. Introduction to the special issue on normative multiagent systems. Autonomous Agents and Multi-Agent Systems, 17(1):1–10. 2008.

Perspectives on Organization in MAS

Organization: purposeful **supra-agent pattern** of **pre-defined** or **emergent cooperation** developed by a designer or by the agents themselves [Boissier et al., 2020].

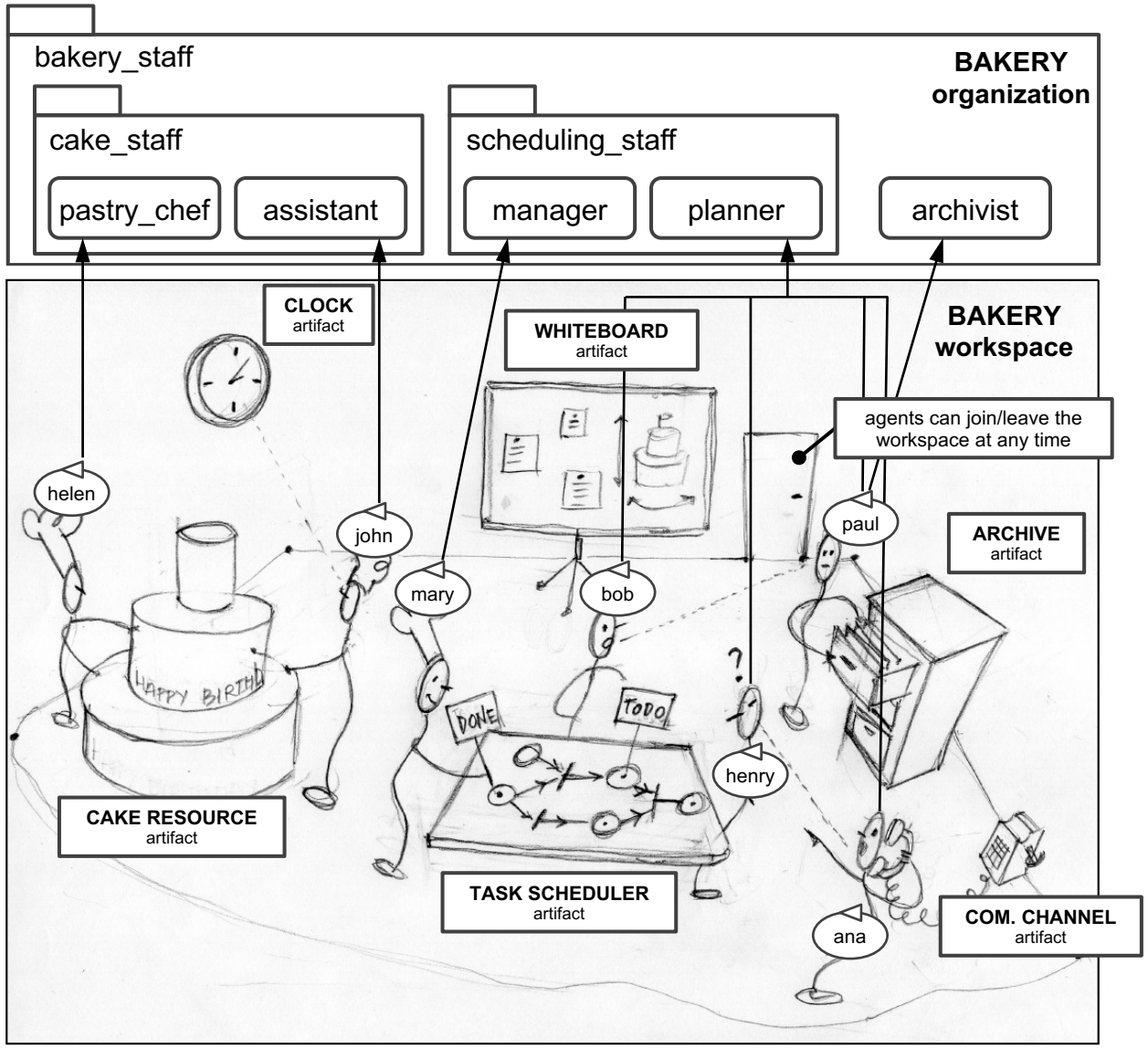


- Organization is observed
- Organization is programmed implicitly
- Organization is a design abstraction
- The org. model is hard-coded
- Organization is observed
- Agents reason to coordinate
- Organization is a prog. abstraction
- Agents reason on the org. model

O. Boissier, *Multi-Agent Programming Course*, MINES Saint-Étienne, 2021.
O. Boissier, R. H. Bordini, J.F. Hubner, A. Ricci. *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*, The MIT Press, 2020.

Normative Organizations: A First Intuition

decouples role of org from role of agents (via indirection)



In **normative organizations**:

- norms are used to coordinate agents towards organizational goals
  - ex: to make a birthday cake
- norms are expressed as part of an organization specification
  - ex: “when making a birthday cake, the **assistant** is **obliged** to prepare the workspace”
- norms are interpreted and considered within the context of an organization entity
  - ex: “to make **this birthday cake** for our client, **John** is now **obliged** to prepare the workspace”

How do **normative organizations** impact the **autonomy** of 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.

Review: Autonomy as a Relational Notion

Autonomy as a Relational Notion

A multifaceted view drawing from cognitive science [Castelfranchi, 1993; Castelfranchi and Falcone, 2003]

An entity X is autonomous from Y about P, where:

- X: the main entity whose autonomy is considered/evaluated
- P: a function/action/goal that must be realized or maintained by the main entity and on which the autonomy is evaluated
- Y: the secondary entity (human, artificial agent, environment, organization, etc.) with respect to whom X should be considered autonomous given the specified function/action/goal P

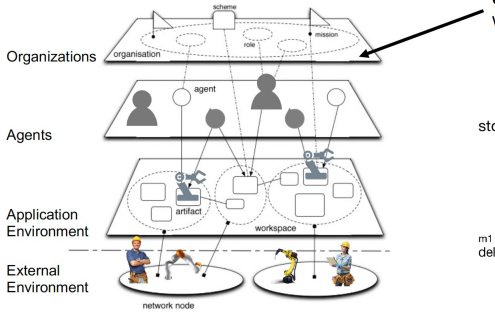
need interdependence for an interactive system

Concrete examples:

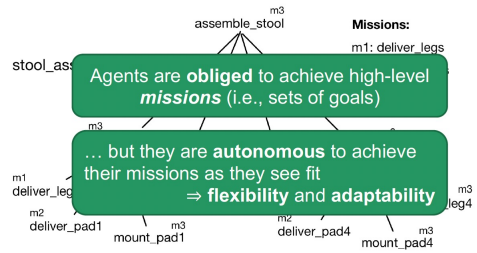
- autonomy ***from other agents***
- autonomy ***from organizations***
- autonomy ***from the environment***

Autonomy from Organizations

"X is permitted/prohibitted/obliged by organization Y to achieve goal G"



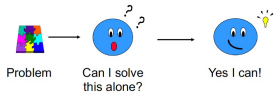
collectives of people and artificial agents working towards shared goals



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

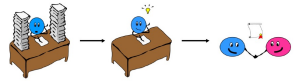
Autonomy from Other Agents

Autonomy as independence



"X is autonomous from (any) Y to achieve goal G"

Autonomy in collaboration



"X depends on Y to achieve goal G" (and knows Y can help achieve goal G)

Independence vs. **Interdependence**

- if agents are fully autonomous from other agents, the MAS is just a set of agents that happen to coexist in a shared environment
- autonomous agents must be able to cooperate, collaborate, compete, and carry complex interactions

O. Boissier, *Multi-Agent Coordination Course*, MINES Saint-Étienne, 2021
C. Castelfranchi, R. Falcone. *From Automaticity to Autonomy: The Frontier of Artificial Agents*. In: Hexmoor H., Castelfranchi C., Falcone R. (eds) Agent Autonomy. Multiagent Systems, Artificial Societies, and Simulated Organizations (International Book Series), vol 7, 2003.

Autonomy from the Environment

- the degree to which an agent's behavior is determined or constrained by the environment

"X is autonomous from the environment to reach destination D"

Applies to **virtual environments** as well!

Industrial robot on a robot-transfer unit (RTU)

Line-follower robot

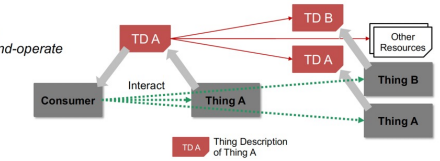
How much is the agent **coupled** to the environment?

Eg TD: Decoupling from environment -> autonomy

Autonomy from the Environment

- the degree to which an agent's behavior is determined or constrained by the environment

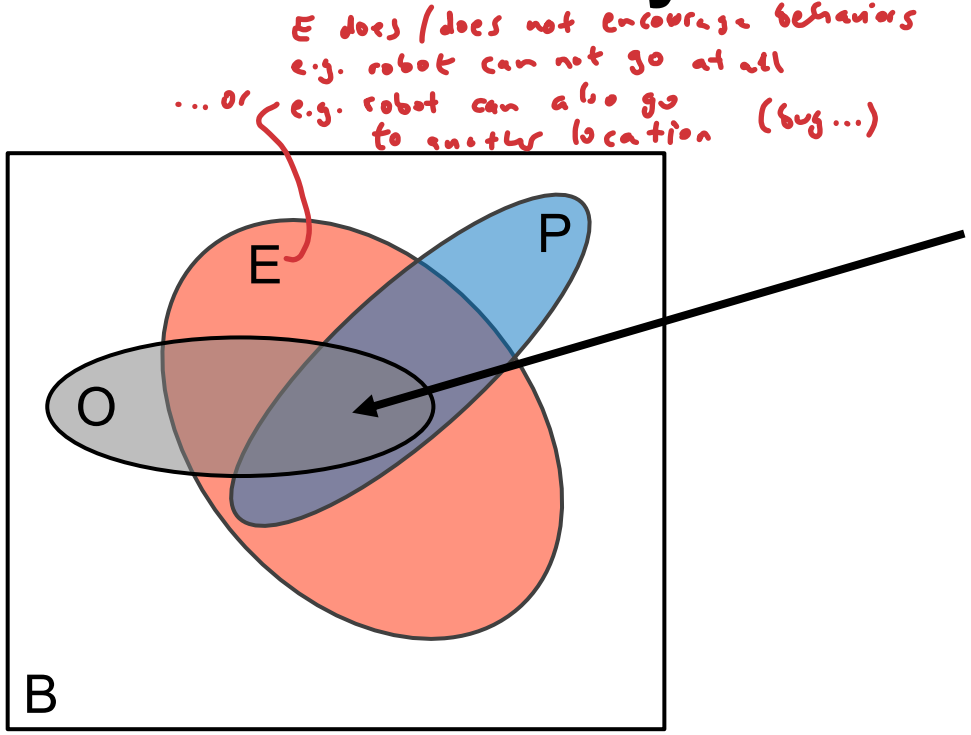
Ideally:
arrive-and-operate



Matthias Kovatsch et al. (eds.), Web of Things (WoT) Architecture, W3C Recommendation, 2020.

How much is the agent **coupled** to the environment?

Autonomy and Organizations: A First Intuition



Desirable outcome:

- $P \cap E \cap O$ not too big
  - **constrains** the agents' autonomy
  - increases **performance**
- $P \cap E \cap O$ not too small
  - **preserves** the agents' autonomy
  - increases **flexibility/adaptability**

leave enough room to achieve design objective with autonomy

- B: agents' possible behaviors in the MAS
- P: agents' behaviors that lead to some global design objective(s) of the MAS
- E: agents' possible behaviors given environment constraints
- O: agents' possible behaviors given constraints from a normative organization

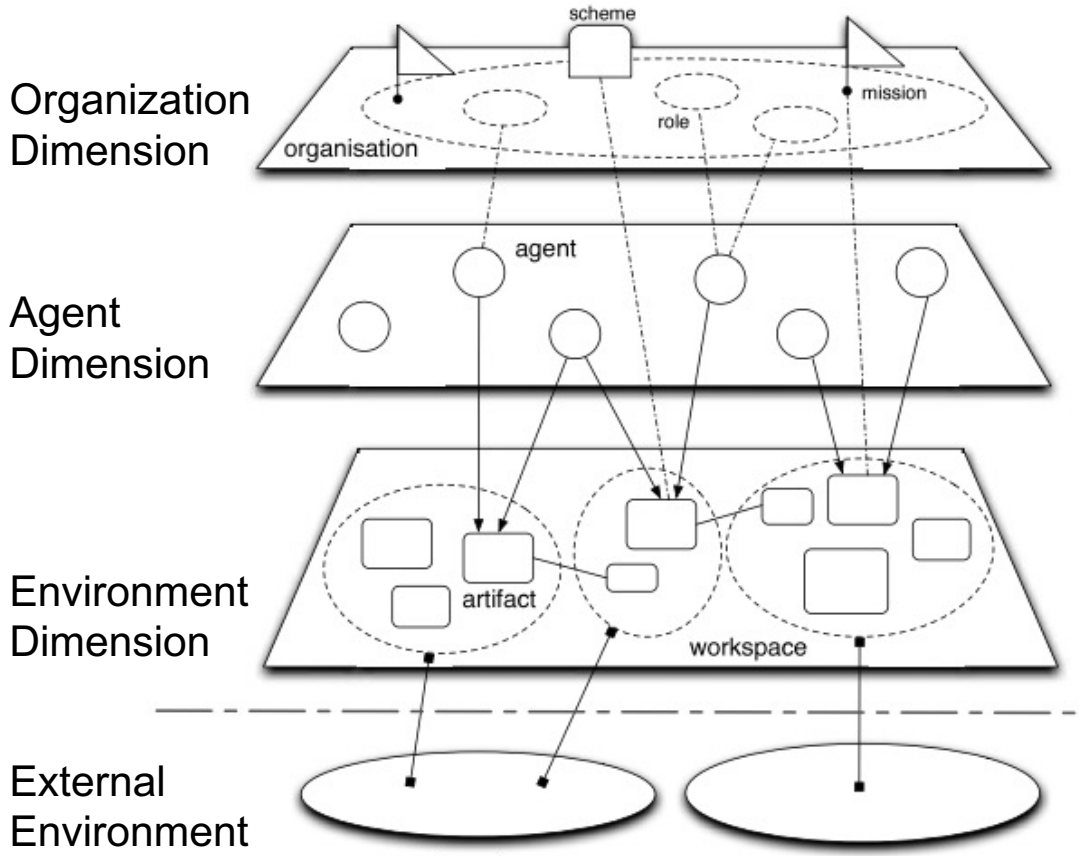
O. Boissier, *Multi-Agent Programming Course*, MINES Saint-Étienne, 2021.

Universität St.Gallen

Today's Agenda

- Normative Multi-Agent Systems
- Normative Organizations
  - Organization-Oriented Programming
  - The Moise Organization Model
  - Hands-on: Programming Organizations with JaCaMo

Engineering Dimensions for MAS



JaCaMo meta-model [Boissier et al., 2013]

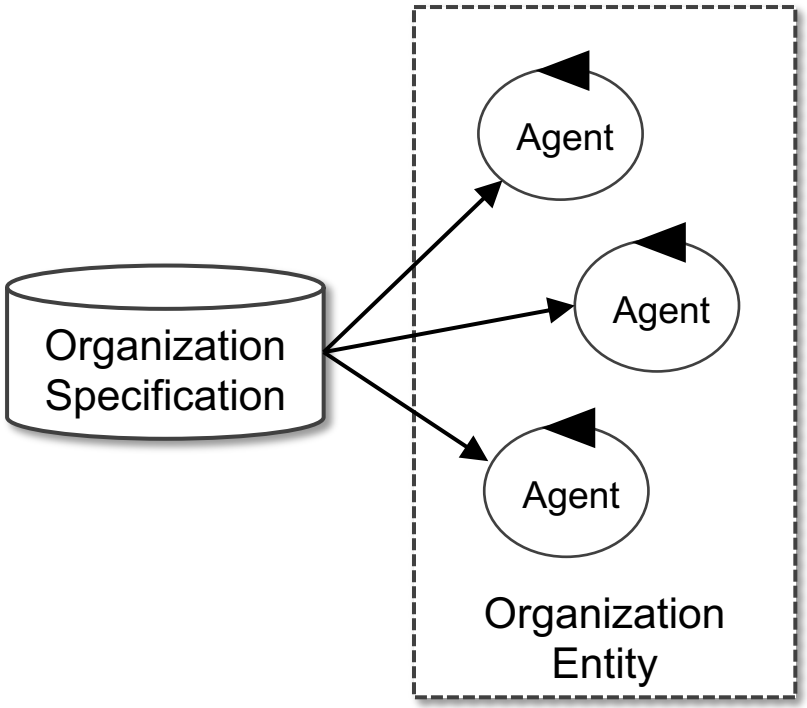
Programming MAS = Programming **Agents**
+ Programming the **Environment**
+ Programming the **Organization**

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

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

Olivier Boissier, Rafael H. Bordini, Jomi F. Hübner, Alessandro Ricci, Andrea Santi, Multi-Agent Oriented Programming with JaCaMo, 2013.

Organization-Oriented Programming



Organization specification:

- defines a cooperative pattern using **organizational concepts** such as roles, groups, workflows, norms, etc.
- a **first-class abstraction**: programmed outside of the agents and outside of the environment
- by changing the organization specification, we can change the **overall behavior** of the MAS

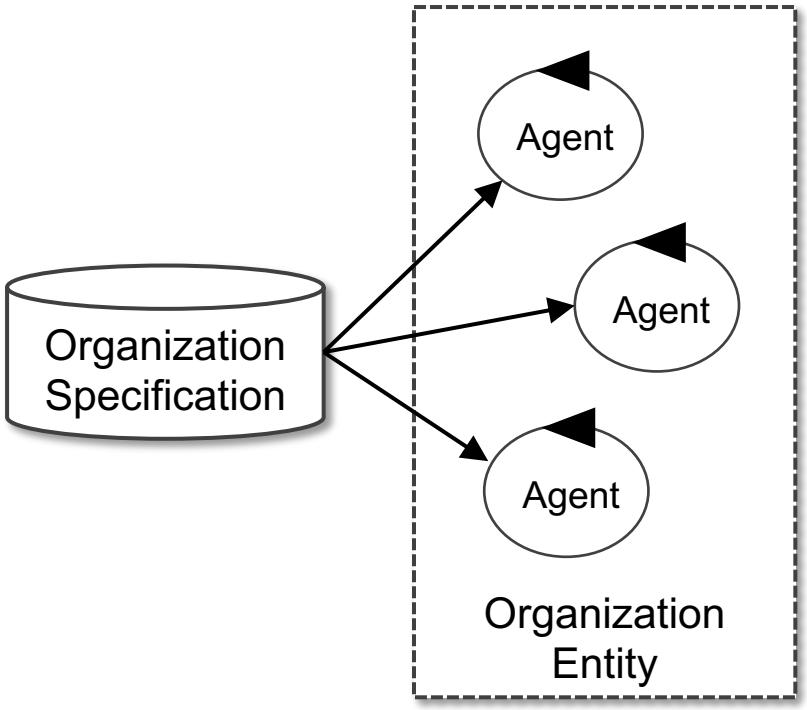
Organization entity:

- an enactment of an organization specification
- can be represented in the mind of agents, and/or can be represented explicitly in the MAS

Program = Organization Specification

O. Boissier, *Multi-Agent Programming Course*, MINES Saint-Étienne, 2021.

Organization-Oriented Programming



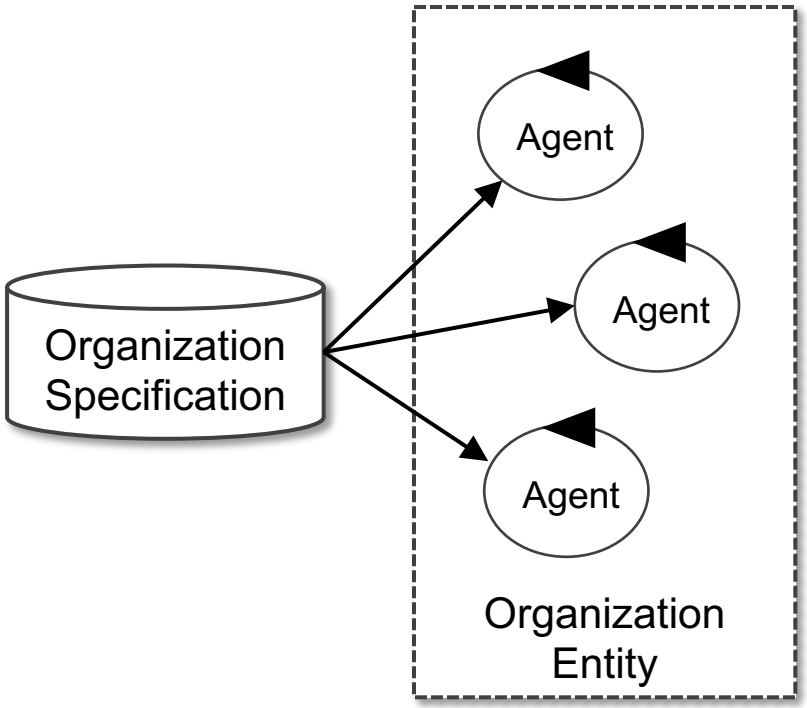
The organization specification is written in a programming language (**organization modelling language**)

Agents **read** the organization specification, **reason** on it, and **follow** the specification (or not)

How to ensure that agents follow an organization specification?

O. Boissier, *Multi-Agent Programming Course*, MINES Saint-Étienne, 2021.

Organization-Oriented Programming

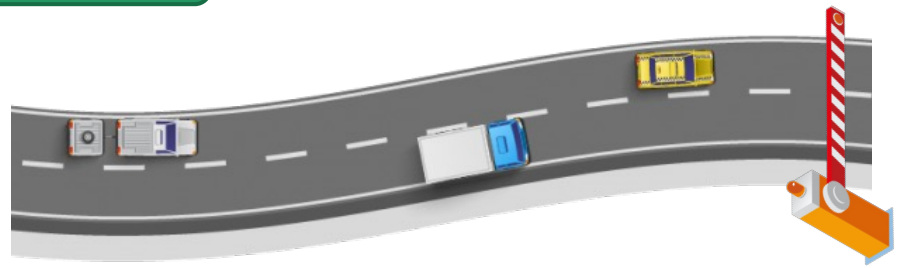


The organization specification is written in a programming language (**organization modelling language**)

Agents **read** the organization specification, **reason** on it, and **follow** the specification (or not)

- **regimentation**: agents are **forced** to follow the specification

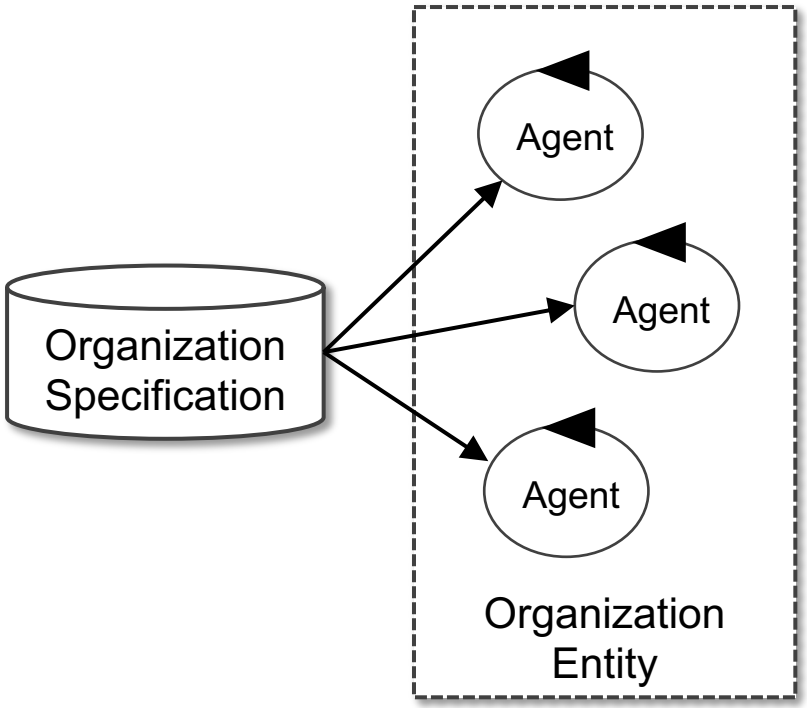
Lecture #1



Objective: design a system for collecting highway taxes

O. Boissier, *Multi-Agent Programming Course*, MINES Saint-Étienne, 2021.

Organization-Oriented Programming



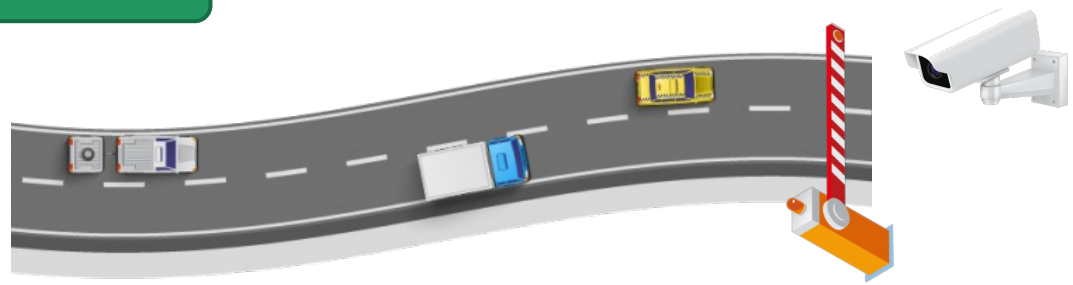
The organization specification is written in a programming language (**organization modelling language**)

Agents **read** the organization specification, **reason** on it, and **follow** the specification (or not)

- **regimentation**: agents are **forced** to follow the specification
- **regulation**: agents are **rewarded** if they follow the specification and/or **sanctioned** if not

Advantages and disadvantages?

Lecture #1



Objective: design a system for collecting highway taxes

O. Boissier, *Multi-Agent Programming Course*, MINES Saint-Étienne, 2021.

Universität St.Gallen

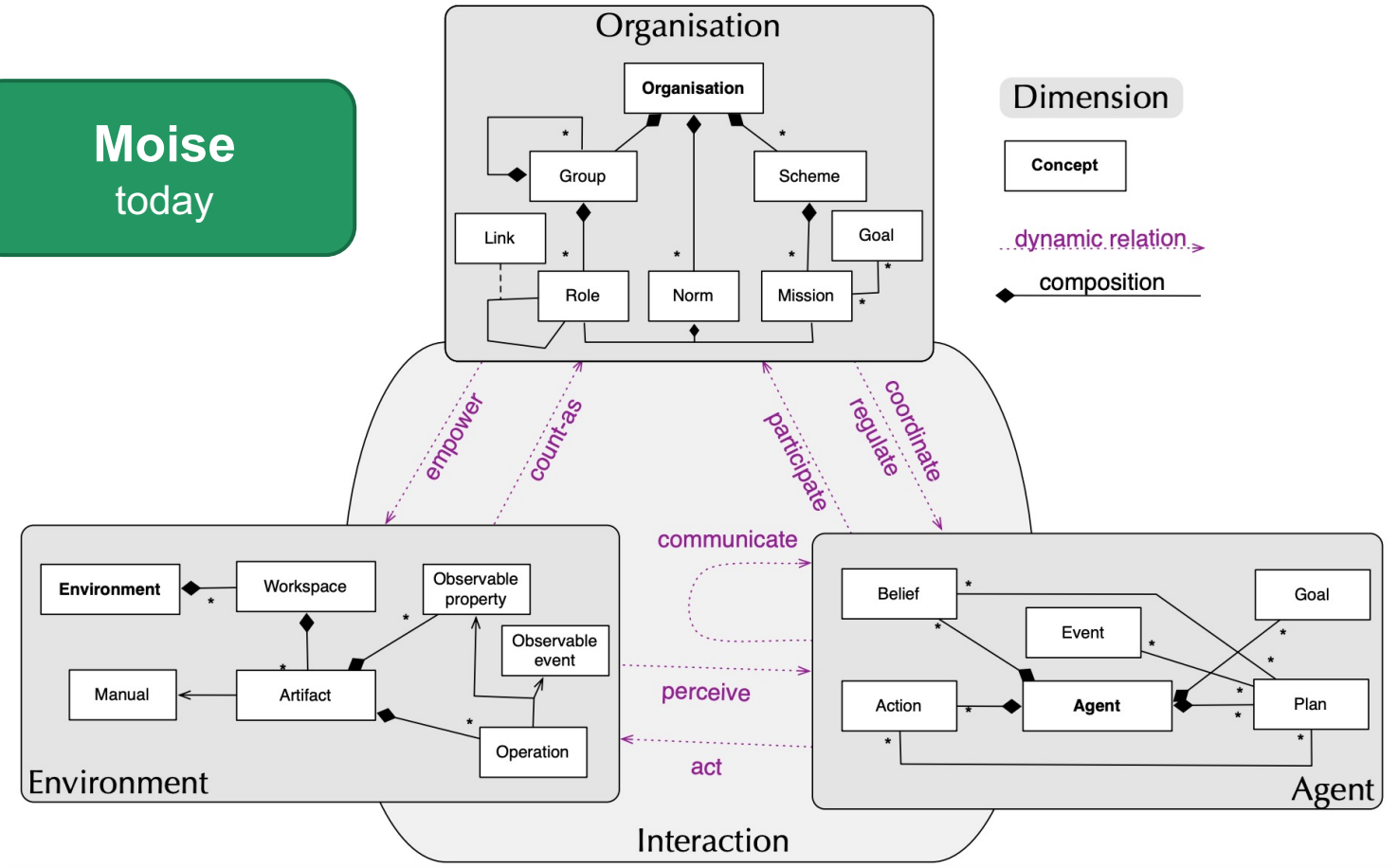
Today's Agenda

- Normative Multi-Agent Systems
- Normative Organizations
  - Organization-Oriented Programming
  - The Moise Organization Model
  - Hands-on: Organization-Oriented Programming with JaCaMo

The JaCaMo Meta-Model Revisited

Moise
today

CArtAgO
Lecture #6



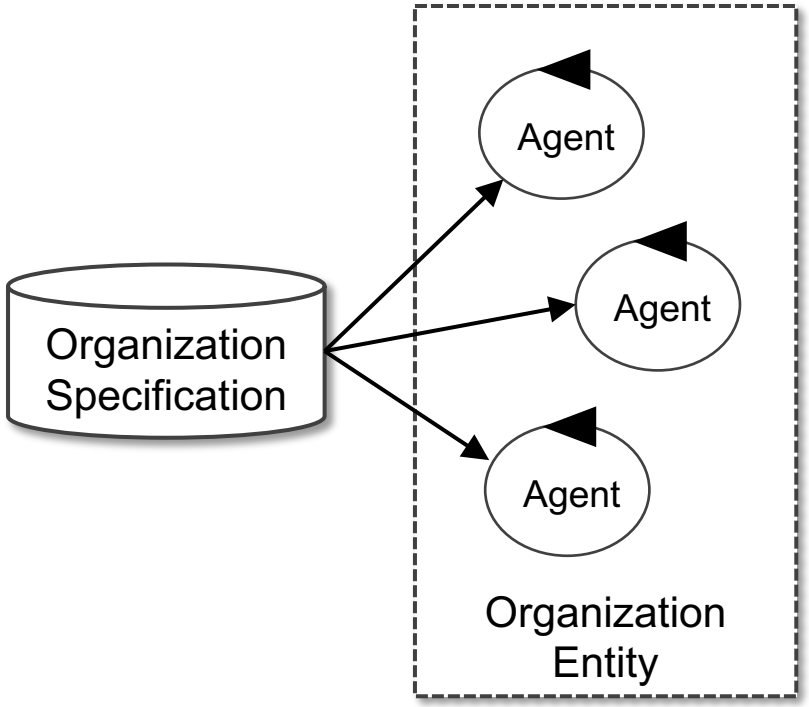
Jason
Lecture #5

JaCaMo = **Ja**son + **CA**rtAgO + **Mo**ise

O. Boissier, *Multi-Agent Programming Course*, MINES Saint-Étienne, 2021.

Programming Organizations in JaCaMo

So far, we've looked at **organizations** as first-class programming abstractions

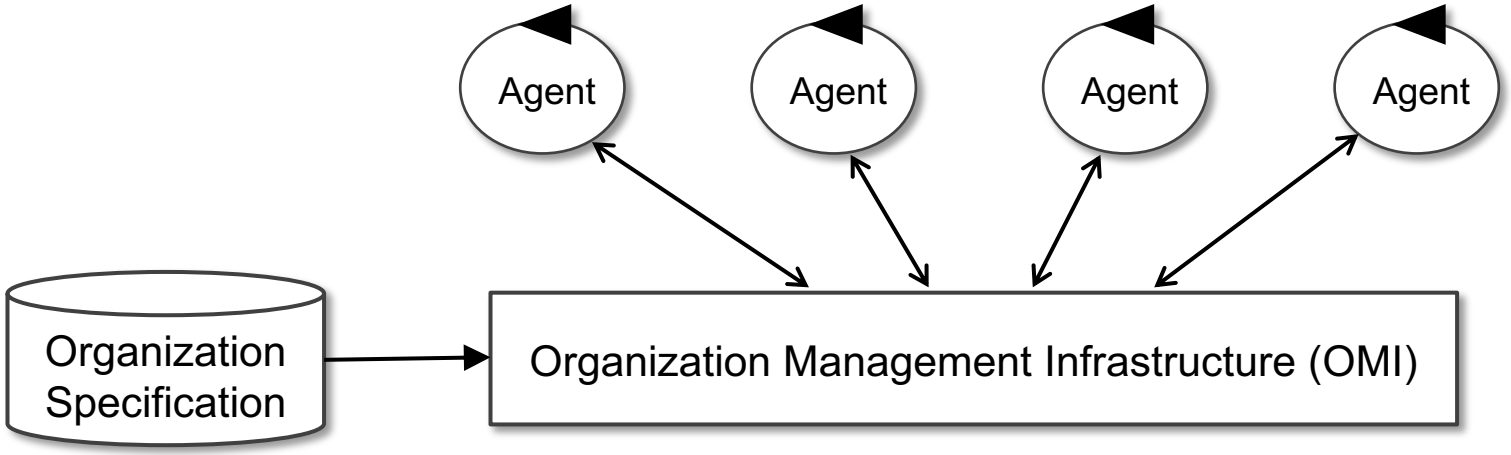


O. Boissier, *Multi-Agent Programming Course*, MINES Saint-Étienne, 2021.

Programming Organizations in JaCaMo

So far, we've looked at **organizations** as first-class programming abstractions

… but organizations are typically managed through a **shared infrastructure**



OMI Responsibility

To **manage the agents' execution** within an organization entity defined by an organization specification

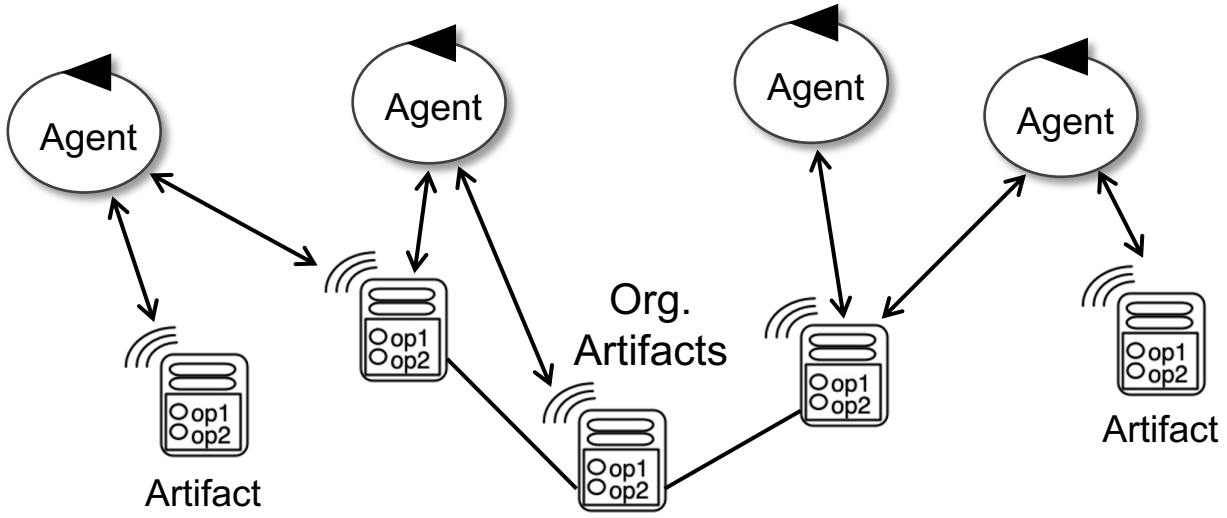
- coordination, regimentation, regulation

O. Boissier, *Multi-Agent Programming Course*, MINES Saint-Étienne, 2021.

Programming Organizations in JaCaMo

So far, we've looked at **organizations** as first-class programming abstractions

… but organizations are typically managed through a **shared infrastructure**



In JaCaMo, the OMI is distributed across a set of organizational artifacts

The agents' working environment is **instrumented** with **organizational artifacts**

- agents **create**, **handle**, and **use** org. artifacts (deploy and manage their OMI)
- **perceive** the organization state and violations of norms
- **decide** about organizational actions, sanctions to apply, etc.

OMI Responsibility

To **manage the agents' execution** within an organization entity defined by an organization specification

- coordination, regimentation, regulation

Cognitive Stigmergy

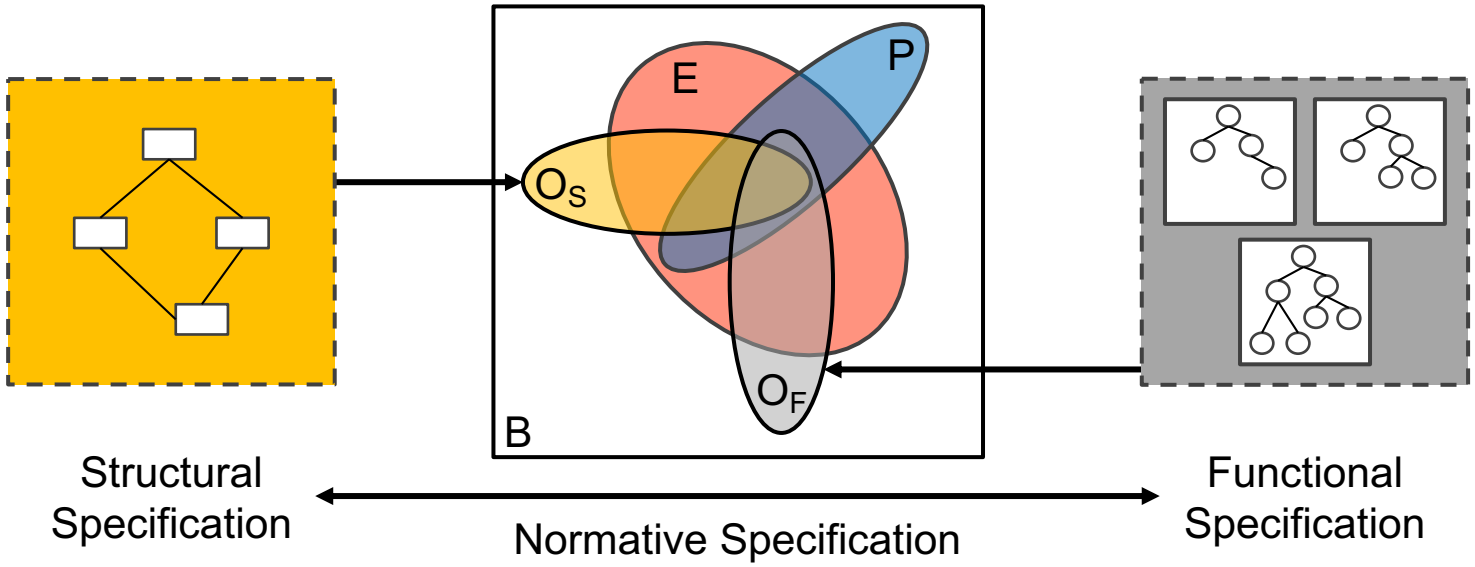
Features supported by the **Moise framework**

O. Boissier, *Multi-Agent Programming Course*, MINES Saint-Étienne, 2021.

The Moise Framework

Moise defines an **organization specification (OS)** on 3 independent dimensions:

- a **structural specification**: defines structure in terms of **roles** and **groups**
- a **functional specification**: defines workflows in terms of **goals**, **missions**, and **schemes**
- a **normative specification**: flexible coupling between the structural and functional specifications through **norms** that assign missions to roles (**permissions** and **obligations**)



O. Boissier, *Multi-Agent Programming Course*, MINES Saint-Étienne, 2021.

The Moise Framework

A Moise OS is defined using the **Moise Organization Modelling Language (OML)**, an XML-based language

To create an organization, agents can use the OS to instantiate an `OrgBoard` artifact

```
makeArtifact(my_auction_org,"ora4mas.nopl.OrgBoard",["src/org/auction-os.xml"], OrgArtId);
```

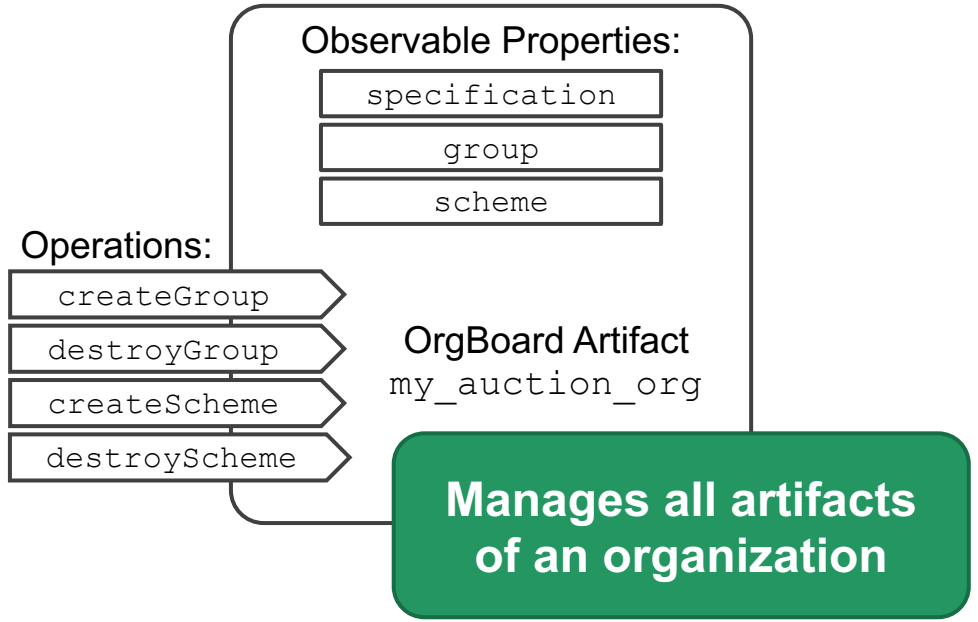
artifact name

points to an OS

new artifact id (output)

OS template in Moise OML

```
<organisational-specification
  <structural-specification>
    ...
  </structural-specification>
  <functional-specification>
    ...
  </functional-specification>
  <normative-specification>
    ...
  </normative-specification>
</organisational-specification>
```



The Structural Specification (Simplified View)

Main concepts: **Role**, **Link**, **Group**

- **Role**: label used to assign constraints on the behavior of agents (an indirection level)
  - a role can **inherit** from one or multiple roles

Example

```
<structural-specification>
  <role-definitions>
    <role id="auctioneer" />
    <role id="participant" />
    <role id="vip_participant">
      <extends role="participant" />
    </role>
  </role-definitions>
  <group-specification id="auction_group">
    <roles>
      <role id="auctioneer"  min="1" max="1"/>
      <role id="participant" min="0" max="300"/>
    </roles>
    <links>
      <link from="auctioneer" to="participant"
      type="communication" scope="intra-group"
      extends-subgroups="true" bi-dir="true" />
    </links>
  </group-specification>
</structural-specification>
```

The Structural Specification (Simplified View)

Main concepts: **Role**, **Link**, **Group**

- **Role**: label used to assign constraints on the behavior of agents (an indirection level)
  - a role can **inherit** from one or multiple roles
- **Link**: relation between roles that constraints the interaction between agents enacting the roles
  - link types: *acquaintance*, *communication*, *authority*

Example

```
<structural-specification>
  <role-definitions>
    <role id="auctioneer" />
    <role id="participant" />
    <role id="vip_participant">
      <extends role="participant" />
    </role>
  </role-definitions>
  <group-specification id="auction_group">
    <roles>
      <role id="auctioneer"  min="1" max="1"/>
      <role id="participant" min="0" max="300"/>
    </roles>
    <links>
      <link from="auctioneer" to="participant"
      type="communication" scope="intra-group"
      extends-subgroups="true" bi-dir="true" />
    </links>
  </group-specification>
</structural-specification>
```

The Structural Specification (Simplified View)

Main concepts: **Role**, **Link**, **Group**

- **Role**: label used to assign constraints on the behavior of agents (an indirection level)
  - a role can **inherit** from one or multiple roles
- **Link**: relation between roles that constraints the interaction between agents enacting the roles
  - link types: *acquaintance*, *communication*, *authority*
- **Group**: a set of roles and relations among roles used to define a shared context for agents
  - groups can contain **sub-groups**
  - groups can define **cardinality restrictions** on roles (**min** and **max** attributes)
  - a group is **well-formed** if the cardinality restrictions of all roles are satisfied

Example

```
<structural-specification>
  <role-definitions>
    <role id="auctioneer" />
    <role id="participant" />
    <role id="vip_participant">
      <extends role="participant" />
    </role>
  </role-definitions>
  <group-specification id="auction_group">
    <roles>
      <role id="auctioneer"  min="1" max="1"/>
      <role id="participant" min="0" max="300"/>
    </roles>
    <links>
      <link from="auctioneer" to="participant"
      type="communication" scope="intra-group"
      extends-subgroups="true" bi-dir="true" />
    </links>
  </group-specification>
</structural-specification>
```

The GroupBoard Artifact

Manages an instance of a group in the organization entity

- exposes the state of the group via observable properties
- exposes operations for participating in the group

Agents can instantiate a `GroupBoard` artifact via an `OrgBoard` artifact using the `createGroup` operation:

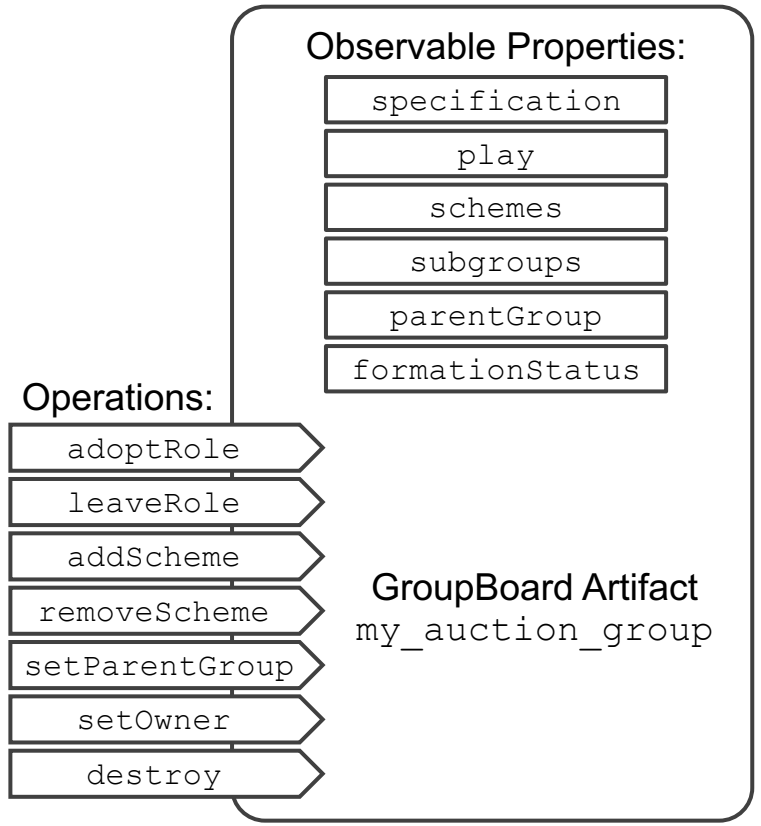
```
createGroup(my_auction_group, auction_group, GrpArtId);
```

- `my_auction_group`: name of the artifact to be created
- `auction_group`: group id in the OS
- `GrpArtId`: id of the artifact to be created (output parameter)

or they can instantiate the artifact directly in their workspace:

```
makeArtifact(my_auction_group,"ora4mas.nopl.GroupBoard",
    ["src/org/auction-os.xml"], GrpArtId);
```

- `"src/org/auction-os.xml"`: points to a local OS



The Functional Specification (Simplified View)

Main concepts: **Goal**, **Mission**, **Scheme**

- **Goal**: organizational objectives; can be *achievement goals* (default type) or *maintenance goals*

Example

```
<functional-specification>
  <scheme id="do_auction">
    <goal id="auction">
      <argument id="Id" />
      <argument id="Service" />
      <plan operator="sequence">
        <goal id="start_auction" />
        <goal id="bid" ttf="10 seconds" />
        <goal id="decide_winner" ttf="1 hour" />
      </plan>
    </goal>
    <mission id="mAuctioneer" min="1" max="1">
      <goal id="start_auction" />
      <goal id="decide_winner" />
    </mission>
    <mission id="mParticipant" min="1" >
      <goal id="bid" />
    </mission>
  </scheme>
</functional-specification>
```

The Functional Specification (Simplified View)

Main concepts: **Goal**, **Mission**, **Scheme**

- **Goal**: organizational objectives; can be *achievement goals* (default type) or *maintenance goals*
  - goals can have **arguments** (key-value pairs)

Example

```
<functional-specification>
  <scheme id="do_auction">
    <goal id="auction">
      <argument id="Id" />
      <argument id="Service" />
      <plan operator="sequence">
        <goal id="start_auction" />
        <goal id="bid" ttf="10 seconds" />
        <goal id="decide_winner" ttf="1 hour" />
      </plan>
    </goal>
    <mission id="mAuctioneer" min="1" max="1">
      <goal id="start_auction" />
      <goal id="decide_winner" />
    </mission>
    <mission id="mParticipant" min="1" >
      <goal id="bid" />
    </mission>
  </scheme>
</functional-specification>
```

The Functional Specification (Simplified View)

Main concepts: **Goal**, **Mission**, **Scheme**

- **Goal**: organizational objectives; can be *achievement goals* (default type) or *maintenance goals*
  - goals can have **arguments** (key-value pairs)
- **Mission**: a coherent set of goals that can be assigned to a role via a norm

Example

```
<functional-specification>
  <scheme id="do_auction">
    <goal id="auction">
      <argument id="Id" />
      <argument id="Service" />
      <plan operator="sequence">
        <goal id="start_auction" />
        <goal id="bid" ttf="10 seconds" />
        <goal id="decide_winner" ttf="1 hour" />
      </plan>
    </goal>
    <mission id="mAuctioneer" min="1" max="1">
      <goal id="start_auction" />
      <goal id="decide_winner" />
    </mission>
    <mission id="mParticipant" min="1" >
      <goal id="bid" />
    </mission>
  </scheme>
</functional-specification>
```

The Functional Specification (Simplified View)

Main concepts: **Goal**, **Mission**, **Scheme**

- **Goal**: organizational objectives; can be *achievement goals* (default type) or *maintenance goals*
  - goals can have **arguments** (key-value pairs)
- **Mission**: a coherent set of goals that can be assigned to a role via a norm
- **Scheme**: a global goal decomposition tree that can be assigned to a (well-formed) group

Example

```
<functional-specification>
  <scheme id="do_auction">
    <goal id="auction">
      <argument id="Id" />
      <argument id="Service" />
      <plan operator="sequence">
        <goal id="start_auction" />
        <goal id="bid" ttf="10 seconds" />
        <goal id="decide_winner" ttf="1 hour" />
      </plan>
    </goal>
    <mission id="mAuctioneer" min="1" max="1">
      <goal id="start_auction" />
      <goal id="decide_winner" />
    </mission>
    <mission id="mParticipant" min="1" >
      <goal id="bid" />
    </mission>
  </scheme>
</functional-specification>
```

The Functional Specification (Simplified View)

Main concepts: **Goal**, **Mission**, **Scheme**

- **Goal**: organizational objectives; can be *achievement goals* (default type) or *maintenance goals*
  - goals can have **arguments** (key-value pairs)
- **Mission**: a coherent set of goals that can be assigned to a role via a norm
- **Scheme**: a global goal decomposition tree that can be assigned to a (well-formed) group
  - any scheme starts from a **root goal** that is decomposed into sub-goals

Example

```
<functional-specification>
  <scheme id="do_auction">
    <goal id="auction">
      <argument id="Id" />
      <argument id="Service" />
      <plan operator="sequence">
        <goal id="start_auction" />
        <goal id="bid" ttf="10 seconds" />
        <goal id="decide_winner" ttf="1 hour" />
      </plan>
    </goal>
    <mission id="mAuctioneer" min="1" max="1">
      <goal id="start_auction" />
      <goal id="decide_winner" />
    </mission>
    <mission id="mParticipant" min="1" >
      <goal id="bid" />
    </mission>
  </scheme>
</functional-specification>
```

The Functional Specification (Simplified View)

Main concepts: **Goal**, **Mission**, **Scheme**

- **Goal**: organizational objectives; can be *achievement goals* (default type) or *maintenance goals*
  - goals can have **arguments** (key-value pairs)
- **Mission**: a coherent set of goals that can be assigned to a role via a norm
- **Scheme**: a global goal decomposition tree that can be assigned to a (well-formed) group
  - any scheme starts from a **root goal** that is decomposed into sub-goals
  - a non-leaf goal is decomposed through **plans** with one of 3 operators: *sequence*, *choice*, *parallel*

Example

```
<functional-specification>
  <scheme id="do_auction">
    <goal id="auction">
      <argument id="Id" />
      <argument id="Service" />
      <plan operator="sequence">
        <goal id="start_auction" />
        <goal id="bid" ttf="10 seconds" />
        <goal id="decide_winner" ttf="1 hour" />
      </plan>
    </goal>
    <mission id="mAuctioneer" min="1" max="1">
      <goal id="start_auction" />
      <goal id="decide_winner" />
    </mission>
    <mission id="mParticipant" min="1" >
      <goal id="bid" />
    </mission>
  </scheme>
</functional-specification>
```

The Functional Specification (Simplified View)

Main concepts: **Goal**, **Mission**, **Scheme**

- **Goal**: organizational objectives; can be *achievement goals* (default type) or *maintenance goals*
  - goals can have **arguments** (key-value pairs)
- **Mission**: a coherent set of goals that can be assigned to a role via a norm
- **Scheme**: a global goal decomposition tree that can be assigned to a (well-formed) group
  - any scheme starts from a **root goal** that is decomposed into sub-goals
  - a non-leaf goal is decomposed through **plans** with one of 3 operators: *sequence*, *choice*, *parallel*
  - schemes can define **cardinality restrictions** on missions (**min** and **max** attributes)
  - a scheme is **well-formed** if the cardinality restrictions of all missions are satisfied

Example

```
<functional-specification>
  <scheme id="do_auction">
    <goal id="auction">
      <argument id="Id" />
      <argument id="Service" />
      <plan operator="sequence">
        <goal id="start_auction" />
        <goal id="bid" ttf="10 seconds" />
        <goal id="decide_winner" ttf="1 hour" />
      </plan>
    </goal>
    <mission id="mAuctioneer" min="1" max="1">
      <goal id="start_auction" />
      <goal id="decide_winner" />
    </mission>
    <mission id="mParticipant" min="1" >
      <goal id="bid" />
    </mission>
  </scheme>
</functional-specification>
```

The SchemeBoard Artifact

Manages an instance of a scheme in the organization entity

- exposes the state of the scheme via observable properties
- exposes operations for participating in the scheme

Agents can instantiate a `SchemeBoard` artifact via an `OrgBoard` artifact using the `createScheme` operation:

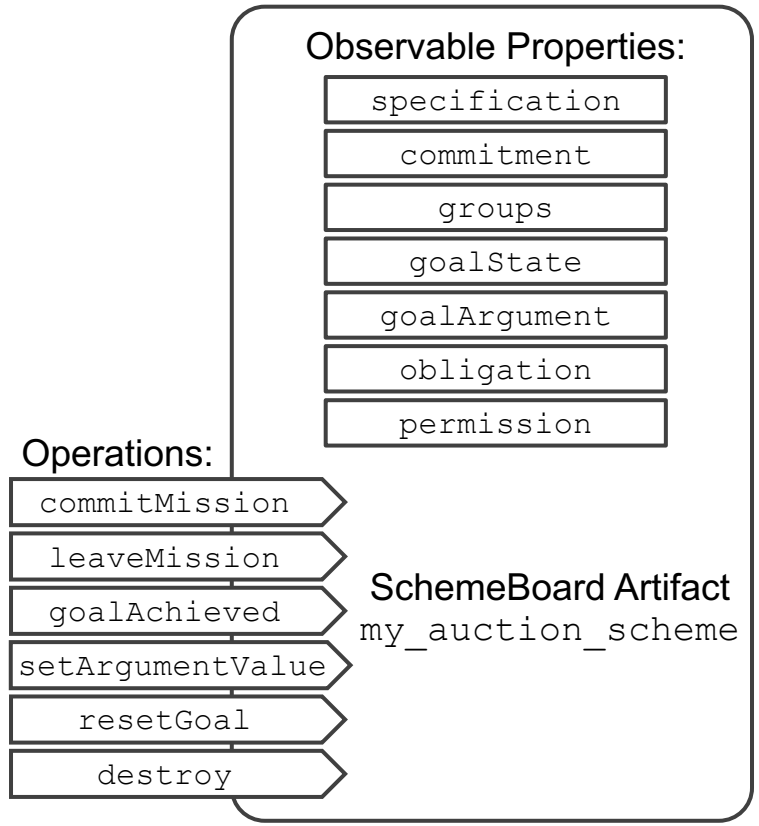
```
createScheme(my_auction_scheme, do_auction, SchArtId);
```

- `my_auction_scheme`: name of the artifact to be created
- `do_auction`: scheme id in the OS
- `SchArtId`: id of the artifact to be created (output parameter)

or they can instantiate the artifact directly in their workspace:

```
makeArtifact(my_auction_scheme,"ora4mas.nopl.SchemeBoard",
    ["src/org/auction-os.xml"], SchArtId);
```

- `"src/org/auction-os.xml"`: points to a local OS



The Normative Specification (Simplified View)

Defines the relation between the structural and functional specifications (**flexible coupling**)

Example

```
<normative-specification>
   <norm id="n1" type="permission"
   role="auctioneer" mission="mAuctioneer" />
   <norm id="n2" type="obligation"
   role="participant" mission="mParticipant" />
 </normative-specification>
```

The Normative Specification (Simplified View)

Defines the relation between the structural and functional specifications (**flexible coupling**)

Main concept:

- **Norm**: a coherent assignment of a mission to a role

Example

```
<normative-specification>
   <norm id="n1" type="permission"
   role="auctioneer" mission="mAuctioneer" />
   <norm id="n2" type="obligation"
   role="participant" mission="mParticipant" />
 </normative-specification>
```

The Normative Specification (Simplified View)

Defines the relation between the structural and functional specifications (**flexible coupling**)

Main concept:

- **Norm**: a coherent assignment of a mission to a role
  - norm types: *permission*, *obligation*
  - any mission-to-role assignment that is not an explicit permission or obligation is a prohibition

The normative specification makes explicit the normative dimension of a role

Example

```
<normative-specification>
   <norm id="n1" type="permission"
   role="auctioneer" mission="mAuctioneer" />
   <norm id="n2" type="obligation"
   role="participant" mission="mParticipant" />
 </normative-specification>
```

Universität St.Gallen

Reasoning About Roles to Adopt

The Moise framework comes with several useful inference rules for reasoning about an OS:

- `role_mission(R, S, M)`: true when there is a norm in the OS that obliges or permits role `R` to commit to mission `M` in scheme `S`
- `mission_goal(M, G)`: true when goal `G` belongs to mission `M`

We can use these 2 rules to write a new rule that determines if a goal `G` is relevant for a role `R`:

```
role_goal(R, G) :-
    role_mission(R, _, M) & mission_goal(M, G).
```

Using the internal action `.relevant_plans`, we can then write a new rule that determines if the agent has plans for achieving a goal `G`:

```
can_achieve (G) :-
    .relevant_plans({+!G[scheme(_)]}, LP) & LP \== [].
```

We can now write a rule that returns true if the agent has plans for all goals relevant to role `R`:

```
i_have_plans_for(R) :-
    not (role_goal(R, G) & not can_achieve(G)).
```

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

Our Journey

Universität St.Gallen

Prerequisites:
- **ASSE**
- **(…)**

Week 1:
Introduction

Week 2:
A Web for Machines

Week 3:
Knowledge Representation and Reasoning for the Web

Week 4:
Linked Data and Distributed Knowledge Graphs

Week 5:
Autonomous Agents (Arch. and Programming)

Week 6 (Coordination I):
Agent Communication and Interaction

Week 7 (Coordination II):
Self-Organization and Stigmergy

Week 8 (Coordination III):
Normative MAS and Organizations

Week 9 (Coordination IV):
Trust & Reputation

Week 10:
Game Theory and Social Choice

Week 11:
Reinforcement Learning and Multi-Agent Learning

Week 12:
An Industry Perspective

Exercises

Ex1: Writing Your First Agent(s)!
Ex2: Automated Planning
Ex3: Web Ontologies
Ex4: Operating on Linked Data
Ex5: BDI Agents
Ex6: Interacting Agents on the Web
Ex7: Ant Colony Optimization
Ex8: Organized Agents
Ex9: Trustworthy Agents
Ex10: Axelrod's Agents
Ex11: Reinforcement Learning Agents
Course Review and Q&A

Universität St.Gallen

Any Questions / Comments / Doubts / Concerns?

Universität St.Gallen

Images

https://www.istockphoto.com/
https://freepik.com