University of St.Gallen

Institute of Computer Science

Web-based Autonomous Systems, FS2023
Danai Vachtsevanou, Andrei Ciortea
{firstname}.{lastname}@unisg.ch

Exercise 8: Organized Agents

Deadline: **May 2, 2023; 23:59 CET**

In this exercise, you will gain hands-on experience in programming multi-agent organizations for coordinating agents. You will implement a Multi-Agent System as a JaCaMo application where:

1.) organizations and organizational artifacts are used as first-class abstractions;

2.) agents join, reason about and cooperate with each other within an organization towards achieving goals without your intervention.

Specifically, you will implement a MAS where agents pursue organizational goals within a MOISE organization for monitoring the conditions in the lab of the Interactions group. The organization operates based on a specification (`org-spec.xml`[1]) that defines an explicit coordination pattern among agents toward keeping individuals in the Interactions lab informed about the current outside temperature. The specification is composed of the following:

- A **structural specification** that defines a group `monitoring_team` with two roles: `temperature_reader` and `temperature_manifestor`. These roles can be played by the agents when participating in a group of this type.
- A **functional specification** that defines and plans the performance goals `read_temperature` and `manifest_temperature` to be achieved by the agents. It structures them into the missions `temperature_reading_mission` and `temperature_manifesting_mission`, respectively. When achieved in a sequence, the two goals contribute to achieving the goal `monitor_temperature`.
- A **normative specification** that assigns obligations to each role. Agents adopting the role `temperature_reader` have the obligation to commit to the mission `temperature_reading_mission`. Similarly, agents adopting the role `temperature_manifestor` have the obligation to commit to the mission `temperature_manifesting_mission`.

① Task 1 (3 points): Initializing Organizations

In the MAS you will be developing in this exercise, an organization agent (`org_agent.asl`[2]) is responsible for initializing organizations. Specifically, the agent will use the specification of the lab monitoring organization[1] to create and interact with essential organizational artifacts that manage the state of the new organization. The artifacts become available in a workspace so that other agents can focus on and use them when joining the organization.

Your first task is to complete the implementation of the organization agent so that it successfully initializes the organization and relevant organizational artifacts. Update the agent program so that the organization agent achieves the following once initialized (Visit the exercise README[3] for more hints and links to useful source code and documentation.):

[1]Organization specification: https://github.com/HSG-WAS-SS23/exercise-8/blob/main/src/org/org-spec.xml.
[2]Organization agent: https://github.com/HSG-WAS-SS23/exercise-8/blob/main/src/agt/org_agent.asl.
[3]Exercise README: https://github.com/HSG-WAS-SS23/exercise-8/blob/main/README.md.

1.) The agent creates and joins an organization *workspace* which is named under the organization name "lab_monitoring_org".

2.) The agent creates and focuses on an *Organization Board artifact*(`ora4mas.nopl.OrgBoard`[4]) within the organization workspace. By focusing on this artifact, an agent (including the `org_agent`) is able not only to manage the organization through the artifact's operations, but also to observe the artifact's observable properties and events that relate to the state of the organization.

3.) The agent uses the Organization Board artifact to create, and then focus on the following organizational artifacts:

- A *Group Board artifact* (`ora4mas.nopl.GroupBoard`[5]) that can be used to manage a "monitoring_team" instance as defined in the specification.
- A *Scheme Board artifact* (`ora4mas.nopl.SchemeBoard`[6]) that can be used to manage a "monitoring_scheme" instance as defined in the specification.

4.) After initializing the organizational artifacts, the agent broadcasts that a new organization is deployed under the name "lab_monitoring_org" with the goal to notify other agents that might decide to join the organization and the monitoring team.

5.) The agent waits until the Group Board artifact signals that a monitoring team has been well-formed, i.e. that all the constraints on the minimal number of agents playing a role are satisfied. After receiving the signal, the agent uses again the Group Board artifact to make the monitoring team responsible for the created monitoring scheme.

② Task 2: Reasoning About the Organization (6 points)

Agents that do not have pre-defined knowledge about the organization will need to reason on the organization specification in order to adopt roles and join groups based on the goals that they can actually achieve (see *Reasoning for Role Adoption*, below). Another agent, unable to reason on the organization, may remain inactive until the organization agent explicitly invites it to adopt a specific role (see *Reasoning on Group Formation*, below).

After you have implemented all steps of this task, you should observe that an agent has adopted the role `temperature_reader`, and another agent has adopted the role `temperature_manifestor`. Due to the `monitoring_team` now being well-formed, you should then observe that both agents directly (i.e., without your intervention) commit to the `temperature_monitoring_scheme` via the relevant Scheme Board artifact, and achieve the goals `read_temperature` and `manifest_temperature` for accomplishing their mission.

- **Reasoning for Role Adoption**

  A sensing agent (`sensing_agent`[7]) in the lab is able to read the outside air temperature because its plan library contains a relevant plan. Even though this agent does not hold any pre-defined knowledge about the organization, it may reason on the organization specification and adopt the role of the `temperature_reader` based on this reasoning, since this role is useful for a mission that requires satisfying the goal of reading the temperature.

  Your task is to update the agent program of the sensing agent so that it behaves as follows (Visit the exercise README[3] for more hints and links to useful source code and documentation.):

[4] Organization Board artifact specification: http://moise.sourceforge.net/doc/api/ora4mas/nopl/OrgBoard.html.

[5] Group Board artifact specification: http://moise.sourceforge.net/doc/api/ora4mas/nopl/GroupBoard.html.

[6] Scheme Board artifact specification: http://moise.sourceforge.net/doc/api/ora4mas/nopl/SchemeBoard.html.

[7] Sensing agent: https://github.com/HSG-WAS-SS23/exercise-8/blob/main/src/agt/sensing_agent.asl.

1.) The agent is able to react to events about new organizations being deployed. The agent reacts a) by joining the deployed organization, and b) by becoming able to observe properties and events that relate to the organization. After implementing b), inspect the sensing agent's belief base to study the agent's beliefs that relate to the organization. By holding such beliefs, the agent is now able to reason about the organization.

2.) Once the agent joins an organization, it reasons on the organization and adopts all its relevant roles. A relevant role is a role that relates to the goals that the agent can achieve by using available plans. By the end of its reasoning, the agent is expected to adopt the `temperature_reader` role, since the agent can use a plan `read_temperature` in the `temperature_reading_mission`.

- **Reasoning on Group Formation**

  The organization agent is responsible for checking whether groups of deployed organizations are well-formed. For this, the agent reasons on an organization specification and the current state of an organization instance. Based on its reasoning, the agent periodically infers whether a group is not well-formed. In that case, the agent invites other agents of the MAS to adopt an available role. An acting agent (acting_agent[8]) in the lab is able to adopt the role.

  Your task is to update the agent programs of the organization agent and the acting agent so that they behave as follows (Visit the exercise README[3] for more hints and links to useful source code and documentation.):

  1.) The organization agent does not simply wait for the monitoring team to become well-formed, but it actively strives for the group to be joined by other agents. Specifically, every 15 seconds the agent infers whether there exist any roles for which the team does not have enough players. For every such role, the agent broadcasts that there is an available role along with the role name and the organization name. After the first 15 seconds, it is expected that the role `temperature_reader` has been adopted, but the role `temperature_manifestor` remains available. This behavior should continue until the group becomes well-formed.

  2.) The acting agent is able to react to events about available roles to be adopted. The agent reacts a) by joining the deployed organization, and b) by becoming able to observe properties and events that relate to the organization. Additionally, the agent c) adopts the available role. It is expected that the agent will end up adopting the role `temperature_manifestor`.

(3) **Task 3: Organization Insights (1 point)** Your third task is to reflect on normative MAS and organizations. Reply to the following question in a short report (max. 1 page of text, the more concise the better):

- Consider that agents might be required to solve (complex) planning problems to reach their goals (remember our session on automated planning). What advantages do you see for having a top-down explicit specification of agent coordination patterns in this case?

(4) **Hand-in Instructions (TBU)** By the deadline, hand in via Canvas a **zipfile** that contains:

1.) a PDF file that includes the link to the GitHub fork containing **code for Tasks 1, and 2**, and your **report for Task 3**;

2.) any additional instructions for how to run your code (if non-obvious).

[8]Acting agent: https://github.com/HSG-WAS-SS23/exercise-8/blob/main/src/agt/acting_agent.asl.