


```

object MoiseSoccerTeamFunctionalSpecification instanceof
  MoiseOrganizationStructure
{
  initial_beliefs:
    (current.describesMoiseFunctionalStructureOf = SoccerTeam);
    (SoccerTeamAttackScheme isInstanceOf MoiseSchemeConcept);
    (SoccerTeamAttackScheme hasMoiseMission m1);
    (SoccerTeamAttackScheme hasMoiseMission m2);
    (SoccerTeamAttackScheme hasMoiseMission m3);
    (SoccerTeamAttackScheme.topMoiseGoal = score_a_goal);
    (SoccerTeamAttackScheme.hasPlan plan_for_scoring_a_goal);
    (m1 isInstanceOf MoiseMissionConcept);
    (m1.hasMoiseGoal score_a_goal);
    (m1.hasMoiseGoal get_the_ball);
    (m1.hasMoiseGoal go_towards_the_opponent_field);
    (m1.hasMoiseGoal kick_the_ball_to_agent_committed_in_m2);
    (m1.firstMoiseGoalInMission = get_the_ball);
    (m1.numberOfAgentsToCommit = 1);
    (m2 isInstanceOf MoiseMissionConcept);
    (m2.hasMoiseGoal score_a_goal);
    (m2.hasMoiseGoal be_placed_in_the_middle_field);
    (m2.hasMoiseGoal go_to_the_opponent_back_line);
    (m2.hasMoiseGoal kick_the_ball_to_the_goal_area);
    (m2.firstMoiseGoalInMission = be_placed_in_the_middle_field);
    (m2.numberOfAgentsToCommit = 1);
    (m3 isInstanceOf MoiseMissionConcept);
    (m3.hasMoiseGoal score_a_goal);
    (m3.hasMoiseGoal be_placed_in_the_opponent_goal_area);
    (m3.hasMoiseGoal shot_at_the_opponent_goal);
    (m3.firstMoiseGoalInMission =
      be_placed_in_the_opponent_goal_area);
    (m3.numberOfAgentsToCommit = 1);
    (plan_for_scoring_a_goal isInstanceOf MoisePlanConcept);
    (plan_for_scoring_a_goal.isPlanForMoiseScheme
      SoccerTeamAttackScheme);
    (plan_for_scoring_a_goal.topMoiseGoal = score_a_goal);
    (plan_for_scoring_a_goal.sequenceMoisePlan(1) =
      get_the_ball);
    (plan_for_scoring_a_goal.sequenceMoisePlan(2) =
      plan_for_field_placement);
    (plan_for_scoring_a_goal.sequenceMoisePlan(3) =
      kick_the_ball_to_agent_committed_in_m2);
    (plan_for_scoring_a_goal.sequenceMoisePlan(4) =
      go_to_the_opponent_back_line);
    (plan_for_scoring_a_goal.sequenceMoisePlan(5) =
      kick_the_ball_to_the_goal_area);
    (plan_for_scoring_a_goal.sequenceMoisePlan(6) =
      shot_at_the_opponent_goal);
    (plan_for_field_placement.isInstanceOf MoisePlanConcept);
    (plan_for_field_placement.isMoiseSubPlanFor =
      plan_for_scoring_a_goal);
    (plan_for_field_placement.topMoiseGoal = score_a_goal);
    (plan_for_field_placement.parallelMoisePlan(1) =
      go_towards_the_opponent_field);
    (plan_for_field_placement.parallelMoisePlan(2) =
      be_placed_in_the_middle_field);
    (plan_for_field_placement.parallelMoisePlan(3) =
      be_placed_in_the_opponent_goal_area);
    ...
}

```

Fig. 12. Brahms code of MOISE⁺ soccer team attack scheme from figure 11

4.2.1) Individual Agent Behavior: We illustrate how agents in our example obtain the code they need to exhibit the behavior appropriate for their role. The Brahms language allows us to model the behavior of our agents from the generic MOISE⁺ behavior, to the domain specific soccer behavior and finally to the individual soccer player agent, depending on the roles these agents play in the organization. In this section we describe how such behavior might be modeled, given the MOISE⁺ structure we discussed so far. In our example, Ajax coach agent Rinus informs agent Sjakie during the training session about the Ajax team organization. Thus agent Sjakie obtains the beliefs that correspond to the MOISE⁺ Ajax organization

as presented in Figure 7³. After agent Rinus talks to agent Sjakie, he goes and talks to both agent Johan and cm_1. Consequently, agent Johan talks to agent Sjakie about his role in the attack scheme. After this agent Johan tells agent Rinus that he spoke with agent Sjakie. Last, agent Rinus tells all the three agents to go and execute the attack scheme. Figure 13 shows agent Rinus talking to agent Sjakie and thus transferring its beliefs. Together this piece of the simulation shows the work practice performance of the Ajax organization, given our scenario. This agent interaction is not part of the MOISE⁺ functional structure as discussed in the previous section. The explanation of the joint agent organization performance of the MOISE⁺ soccer attack scheme comes next.

Figure 14 shows the execution of the soccer attack scheme by the three agents performing mission m1, m2 and m3. Mission m1 is committed to by agent cm_1, m2 by agent Johan and m3 by agent Sjakie. The execution of the scheme starts for each agent, at the same time, with the execution of the workframe *wf_ExecuteMission*. The code for this, and other workframes and activities, can be found in the Appendix. The *ExecuteMission* activity is domain dependent, and is implemented in the domain group *PlayerRole*. This way all soccer player agents will inherit the same activity, and can execute the mission in the same way.

In our example scenario, the only activity that our agents can execute is *PlayingSoccer*. This activity is the main activity that each soccer player agent executes, based on the mission that it is committed to. All sub-activities for playing soccer are defined as workframes inside the *PlayingSoccer* activity (see figure 15). For source code of this activity, see the Appendix.

4.3. Collaboration aspect

The process of collaboration, essential for team work is not defined by MOISE⁺: a) how the three players on a soccer team decide together to dynamically and instantly create a group or sub-team to execute the attack scheme, b) how the players decide who is committing to which mission, and c) what activities the players should perform to obtain the goals associated with their mission. These collaboration aspects are left to the soccer team, or even more specifically, to the individual players of the soccer team.

In order to model a group of players deciding to manage the performance of a MOISE⁺ scheme, we adhere to the *team-work model* as described by Sierhuis, et al. in [41]. In this model, the dynamic creation of a group or team of agents follows five distinct phases (see figure 16). Within the soccer team, using the team-work model, the soccer team members can dynamically form sub-teams to fulfill the team's ultimate goals of scoring more goals than the opposing team.

4.4. Regulation aspect

Regulations in organizations are meant to improve the efficacy of its operations. In human societies norms can be described as the unwritten rules of the organization, in contrast

³Agent Sjakie_Meulemans gets a lot more beliefs than are shown in Figure 7, because agent Rinus_Michels tells agent Sjakie about all the Ajax team agents, whereas Figure 7 only shows the agents relevant for the example.

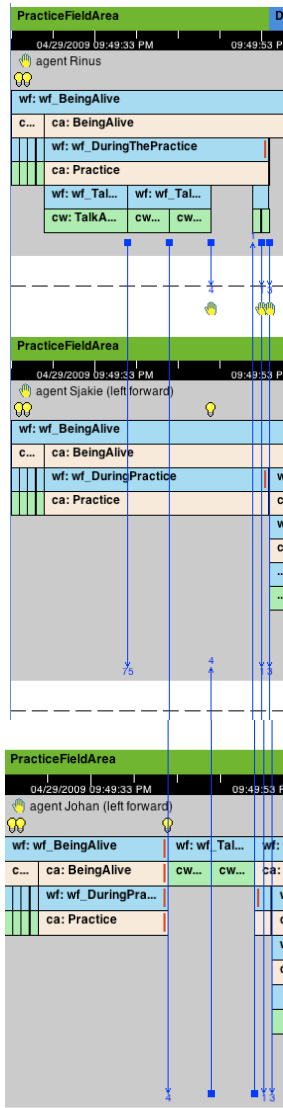


Fig. 13. Brahms sim screenshot of the agent activity timeline for agent Rinus, agent Sjakie, and agent Johan. The vertical (blue) arrows in the screenshot show agent communication over time. The execution of workframes and activities is shown as the (blue, skin and green) colored horizontal bars, while the thoughtframes (reasoning rules) are shown as little lightbulb icons above the activities. The horizontal (green) bar at the top, above the (black) timeline bar, shows the location of the agents (i.e., the PracticeFieldArea). Agent Rinus is first communicating to agent Sjakie during the Practice activity. Next Rinus is talking to agents Johan and cm_1 (agent cm_1 is not shown, but is shown as a little hand icon underneath agent Rinus). Then you can see agent Johan communicating with agent Sjakie, and the with agent Rinus. All this is being done during the execution of the activity Practice.

with the laws and policies that comprise the written rules. As soon as we simulate an organization, having unwritten rules is impossible. Therefore, unwritten has to be translated to “unofficial.” Norms, policies and laws have in common that individuals can decide to violate them. The violation of a norm, if noticed by other agents, can cause the violating agent, for example, to suffer rebukes and/or a loss of reputation. However, violation of a norm cannot be punished by the institution. The violation of policies and laws, if detected, can lead to the same effects as violation of norms, but violation of policies and laws can also be punished by the institution.

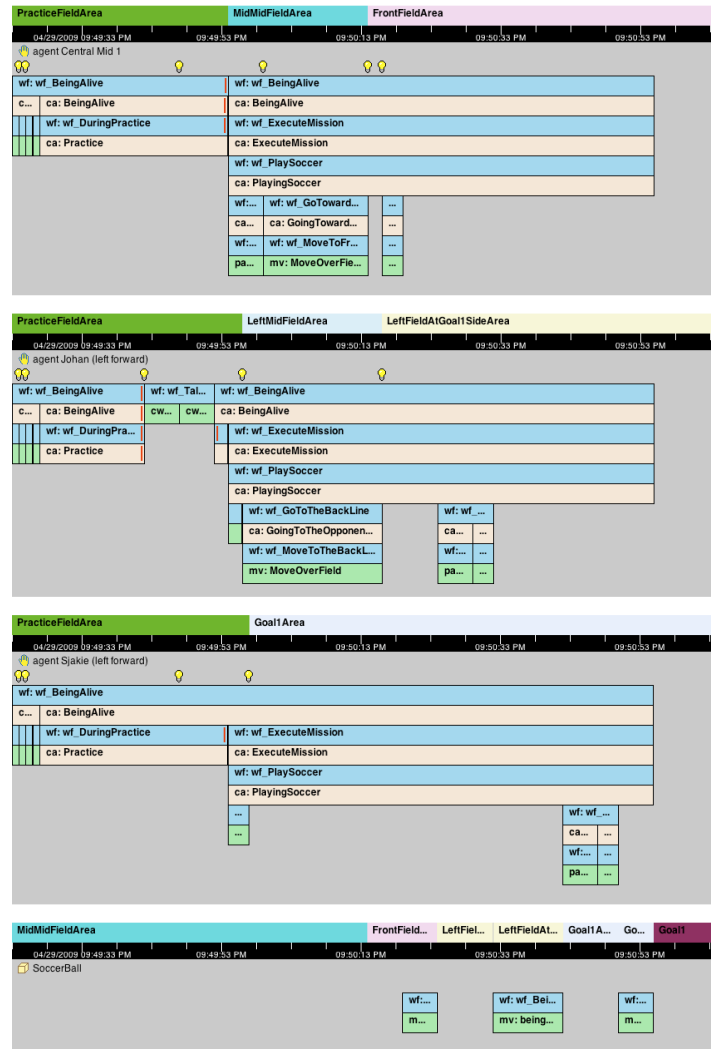


Fig. 14. Brahms sim screenshot of the agent activity timeline for agent Rinus, agent Sjakie, and agent Johan.



Fig. 15. Brahms sim screenshot of the agent activity timeline for agent Rinus, agent Sjakie, and agent Johan.

Phase Number	Teamwork Phase
1	Recognition of the need of help from other agents
2	Team formation
3	Ongoing coordination and team maintenance throughout task execution
4	Recognition of resolution or impasse
5	Team disbanding

Fig. 16. Teamwork phases to dynamically create and dismember teams

