Organizationally Adept Agents

Daniel D. Corkill,[1] Edmund H. Durfee,[2] Victor R. Lesser,[1] Huzaifa Zafar[1], and Chongjie Zhang[1]

[1] University of Massachusetts Amherst, Amherst MA 01003, USA
[2] University of Michigan, Ann Arbor MI 48109, USA

Abstract. An *organizationally adept* agent (OAA) is not only aware that it is part of an agent organization and about its role(s) in that organization, but it can also assess how well it is fulfilling its organizational responsibilities and can proactively adapt its behaviors to meet organizational needs better. OAAs evaluate their behaviors based not only on their (agent-centric) self-interests, but also on their (organization-centric) responsibilities to each other and their (social-centric) willingness to perform activities requested by other agents.

Agent organizations designed for less adept agents must specify detailed guidelines that, when blindly followed, will influence individual agents to work together in the expected environment. However, if the environment deviates from expectations, such detailed organizational guidelines can mislead agents into counterproductive or even catastrophic behaviors. Organizations designed for OAAs, on the other hand, can assume that agents will reason about organizational expectations, and will adjust their behaviors when the nominal guidelines misalign with those expectations.

We describe an extended BDI reasoning architecture for an OAA that balances organizational, social, and agent-centric interests and can adjust this balance when appropriate. Then we show how we used the reasoning architecture with agents operating in RoboCup Rescue scenarios.

Keywords: organizationally situated agents; multi-agent organizations; adaptive agent behavior; emergent agent organizations

1 Introduction: Organizational Structuring and Control

Despite creating highly capable software agents, widespread deployment of large multi-agent applications has not occurred, in large part because coordinating many individual agents' activities to achieve collective benefit in open, dynamic environments is hard. Statistical approaches (like those inspired by insect colonies) allow miscoordination so long as aggregate behavior of the population is acceptable [2], but can waste valuable agent resources. More effective coordination can arise when each agent reasons about the past, present, and future activities of all agents, given the evolving environment state, to make informed decisions. This can work well for small teams and coalitions [17], but quickly becomes intractable in larger agent populations. Avoiding intractability in large-scale settings by pre-specifying what activities each agent should perform cannot cope with an environment that changes unpredictably.

A variety of techniques for scalable coordination have been investigated. In general, these try to reduce the awareness an agent needs of others. For example, agents can exploit *locality* to only be aware of nearby agents, such as where an agent responsible for piloting a robotic truck needs only model other nearby vehicles. Or agents can utilize *aggregation* to model many other agents with a small set of parameters, such as how a product's price quote from an auction can summarize the bidding behaviors of all the buyers and sellers. Underpinning these and other scalable coordination techniques, however, are more stable structuring decisions about permissible interactions (e.g., the "rules of the road") and communication patterns. These structuring decisions in turn can require considerable designer effort and insight to ensure that operational coordination mechanisms (e.g., negotiation protocols and auctions) can perform effectively and tractably. Furthermore, the resulting structures cannot adapt themselves to changing circumstances without human intervention, and might rely instead on compensating (over)use of operational coordination techniques which will either yield lower-quality coordination decisions, incur greater amounts of overhead, or both.

These limitations can be confronted by treating organizational control organically within a multi-agent system rather than relying on an external (human) designer. *Organizational control* is a multi-level approach to coordination in which organizational goals are identified and used in determining agent roles and responsibilities [4]. These roles and responsibilities provide a context for making detailed *operational control* decisions by the individual agents. Organizational control is distinct from operational control in both time span and level of detail. Organizational roles and responsibilities specify general, long-term guidelines that are subject to ongoing elaboration and revision by the agents [6]. Operational control, on the other hand, involves specific short-term agreements among agents to perform specific activities at specific times [12]. Organizational guidelines reduce the complexity of operational decision making, lower the cost of distributed resource allocation and agent coordination, help limit inappropriate agent behavior, and reduce unnecessary communication and agent activities.

Corkill and Lander [5] note that effective multi-agent organization becomes increasingly important with: 1) increases in the number of agents and their diversity; 2) increases in the duration of agent activities; 3) increases in the repetitiveness of agent activities; 4) increases in resource sharing; 5) increases in agent collaboration and the complexity of interactions; 6) increases in agent specialization; 7) decreases in agent capability; and 8) decreases in resource slack (or increases in performance requirements). As agent-based applications become more widespread and complex, effective organizational coordination will become an important aspect of system performance. We are just reaching the point of creating multi-agent applications where organization makes a meaningful difference, and soon it will become the critical difference.

In the next section, we discuss organizationally adept agents in detail, focusing on the reasoning activities that they must perform. In Section 3 we describe a BDI-based architecture that we have developed to enable an agent to perform organizationally adept reasoning. Then we show how we used the reasoning architecture with agents operating in RoboCup Rescue [11] scenarios.

2 Organizationally Adept Agents

Central to effective organizational control is the creation of *organizationally adept* agents (OAAs) that can reorient their local activities based on their interpretation of the organizational intent, allowing emergent organizational behavior within designed organizations. Emergent behavior in multi-agent systems has been the topic of considerable interest and research [3, 14, 1]. We are primarily interested in emergent *organizational* behavior where individual agents identify interaction and local control decisions that have been effective in the past and give preference to similar decisions in the future. Gasser and Ishida were among the first to consider emergent organizational behavior as a means of organization self-design [7]. In their work, each agent used what it knew about how the organization was performing in its local neighborhood to make unilateral changes in its organizational behavior. Kamboj and Decker extended the Gasser and Ishida work to more complex multi-agent environments [10]. We, and other researchers, have used learning and diagnosis techniques to enable agents to determine emergent behavior [16].

An OAA must be able to: a) adapt its behavior given explicit expectation-annotated organizational directives, b) determine appropriate emergent organizational behaviors on its own (perhaps within organizationally delimited boundaries) when organizational expectations are not met or when externally designed directives are absent, and c) report deviations from organizational expectations as well as new emergent organizational behaviors to other agents.

OAAs are especially beneficial in settings where either the system objectives or the task-environment characteristics are not fully known. In addition, OAAs provide mechanisms to adapt (both short and long-term) to changing environmental conditions. However, the benefits provided by OAAs diminishes when the objectives of the system are not identified at all, when an agent cannot generate expectations of its own task performance or measure if it is achieving those expectations, and when agents are unaware of their neighbors and expectations of their performance. The importance of agent expectations (and the OAA approach) does not depend on whether the agents are assembled by negotiated agreements between self-interested agents or whether they are inherently inclined to achieve common social objectives.

2.1 Functional OAA Architecture

Fig. 1 presents the basic functional architecture of an OAA. The agent's organizational behavior derives from a combination of annotated organizational

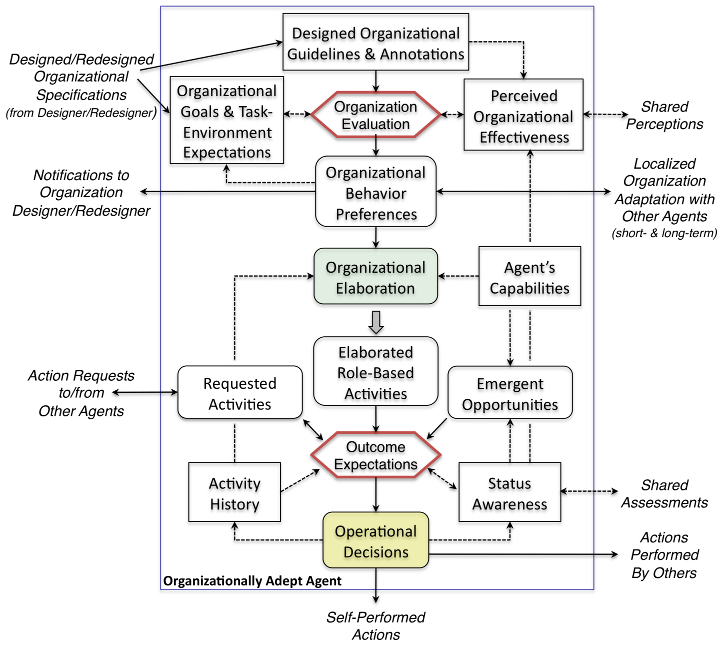

Designed Organizational Guidelines & Annotations
Designed/Redesigned Organizational Specifications (from Designer/Redesigner)
Organizational Goals & Task-Environment Expectations
Organization Evaluation
Perceived Organizational Effectiveness
Shared Perceptions
Notifications to Organization Designer/Redesigner
Organizational Behavior Preferences
Localized Organization Adaptation with Other Agents (short- & long-term)
Organizational Elaboration
Agent's Capabilities
Elaborated Role-Based Activities
Action Requests to/from Other Agents
Requested Activities
Emergent Opportunities
Outcome Expectations
Activity History
Status Awareness
Shared Assessments
Operational Decisions
Actions Performed By Others
Organizationally Adept Agent
Self-Performed Actions

Fig. 1. Organizationally Adept Agent

guidelines that are disambiguated and elaborated into specific actions by the OAA and from actions identified from the agent's local view of its opportunities (which may not align with the organizational guidelines).

The activities shown in the boxes at the top of the figure are associated with organizational control aspects of the OAA. An organization designer provides the OAA with organizational guidelines and performance expectations that are elaborated into specific role-based (organization-centric) operational activity possibilities. Activities to be performed by an OAA can also be requested by other agents, and these social-centric activity possibilities are shown at the lower left side of the figure. On the lower right side are emergent (agent-centric) opportunities stemming from the agent's awareness of its environment, capabilities, and status. Decisions about what organizational, social, and self-centric possibilities should be performed by the agent (or should be requested to be performed by other agents) are made by the "Operational Decisions" function at the bottom center of the figure. This reasoning activity is at the heart of the OAA approach and, after presenting an example of the type of agent behavior that we expect an OAA to perform, we describe a BDI-based implementation of this core OAA functionality.

2.2 An Example of OAA Behavior

Consider an initial organization, shown in Fig. 2, for the RoboCup Rescue domain. As part of this organization, the city is divided into four regions. A *center* is assigned the role of managing each region. The activities of a center include: maintaining information about buildings in its region, collecting status reports (such as location, water levels, etc.) from each fire-brigade agent within the region, determining the number of fire brigades that are needed to extinguish a building on fire, determining which fire brigade agents to assign to which buildings, and requesting help from neighboring centers when required. Each center is provided with organizational guidelines that include the fire-brigade agents that the center manages[3] and parameters for the center's managerial role (for example, the boundary of the region it manages). Additionally, the center's organizational guidelines are annotated with task-environment expectations provided by the organization designer.[4] For example, annotations might specify that the center should typically have to manage at any given time no more than: one medium task (requiring two fire brigades to extinguish) and one small task (requiring one fire brigade); three small tasks; or one large task (requiring 3 fire brigades). Assuming that organization was designed for a city where fires are evenly distributed across regions, an equal number of fire brigades have been assigned to each region.

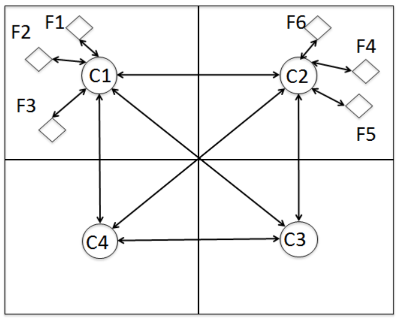
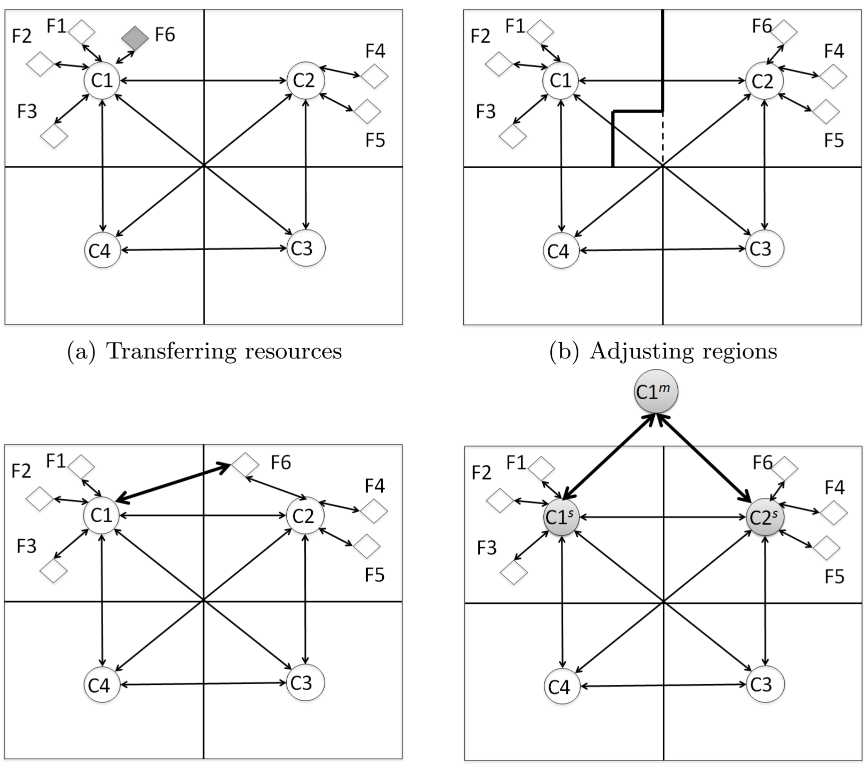


Fig. 2. An example RoboCup Rescue organization where the city is divided into four regions, with a manager (centers C1, C2, C3, and C4) responsible for each region. Agents assigned the role of extinguishing fires (fire brigades F1 to F6) are also shown for the C1 and C2 regions.

Now, consider center C1 operating in the Fig. 2 structure. As new fires are discovered, C1 begins allocating resources (assigning fire brigades the task of extinguishing buildings on fire) to fires in its region in accordance with the parametrized role assigned to it—even though it could potentially perform other activities (such as allocating resources to fires outside its region). Suppose C1 discovers new fires in its region that require more resources than C1 has available. What should C1 do as an OAA? First of all, C1 needs to make operational decisions about how it should address the immediate problem. Should C1 allow new fires to stack up until it can assign its fire-brigade resources to them (hoping that the arrival rate will drop back to manageable levels quickly)? If C1 delays responding to fires too long, buildings might burn down or, worse, the fires might spread to neighboring buildings, compounding the problem. Given this, should C1 delay

[3] Those fire-brigade agents also having been assigned the role of extinguishing fires as directed by the center.

[4] In this paper, we do not care if the agent organization is designed by a human or by an automated (potentially distributed) designer process [15, 9]. In either case, we assume that annotations sharing the designer's expectations are provided along with an agent's organizational guidelines to help an OAA recognize when the assumptions used by the designer become invalid.



(a) Transferring resources (b) Adjusting regions

(c) Sharing resources (d) Employing a higher-level management role

Fig. 3. Adjustments to the organization (from Fig. 2)

dealing with the newly discovered fires? Or, should C1 assign a smaller number of resources to these fires with the goal of preventing them from spreading, rather than extinguishing them? Should C1 ask a neighboring center for help, and if so, which center? If C1 decides to seek help from a neighboring center, should it simply borrow additional resources (and incur an added span-of-control challenge in managing the additional brigades) or should it transfer the responsibility of handling the fire to the neighboring center?

Furthermore, from an organizational-control perspective, C1 needs to consider whether the overload situation is: 1) a transient burst whose effects are so brief that they lie within the environmental expectations that were assumed when the organization was designed; 2) a longer duration event that is outside the organizational expectations (but will return to normal eventually); or 3) a change in the environmental situation from the original assumptions, and the design is no longer appropriate.

In deciding what to do, C1 must generate candidate operational-action choices and consider potential longer-term changes to its organizational behavior. Consider how C1 might proceed under case 2, above, where the longer-term volume of fires in its region exceeds both C1's resource constraints and its organizational expectations. Assume that this situation has become recurring, and that each time it has occurred C1 has addressed it operationally by borrowing resources from a neighboring center. Most of the time, C1 has borrowed brigades from C2, and C1 is now approaching C2 first in order to reduce the operational cost of seeking resources from other centers. The next step in addressing this recurring situation is to move from repeated operational "fixes" to a long-term agreement between C1 and C2.

An OAA makes localized organizational-behavior modifications by obtaining long-term agreements with other OAAs. In doing so, OAAs assess, implement, and evaluate the effect of these behavior modifications locally, and then, if appropriate, report them to the organization designer as information to be used in modifying the overall organization design, if appropriate. As shown in Fig. 3, C1 and C2 can form one of several local long-term agreements:

Transferring resources—If C2 perceives that it generally has a spare fire brigade even when operating near the limit of the task-environment expectations associated with its organizational guidelines, C1 and C2 can agree to permanently transfer control of a fire brigade (e.g., F6) to C1. To accomplish this agreement, C1, C2, and F6 need to form a long-term agreement among themselves and annotate it with the corresponding environment expectations.

Adjusting Regions—If C2 perceives that the long-term task volume is lower than its local processing capability and organizational expectation but sometimes it needs all three fire brigades, C1 and C2 can mutually decide to adjust the region boundaries so that the expected task distribution between the two new regions is in line with organizational expectations.

Sharing Regions—Regions can also be adjusted so that they overlap. Fires in buildings in the overlap area can be handled by either center (providing increased resource flexibility), but at the complexity of having the centers determine who will handle each fire in the overlap area. For example, C1 and C2 could decide to negotiate each time, or simply allow the center with more available resources to take charge.

Sharing Resources—If C2 perceives that the long-term task volume is in line with its local processing capability and organizational expectations and that C1 and C2 rarely require simultaneous use of three or more fire brigades, C1 and C2 can agree to share a fire brigade (e.g., F6). That is, both C1 and C2 can direct F6 and F6 reports its status information to both centers. The agreement with F6 needs to specify how it will deal with conflicts, such as if C1 and C2 request it to fight different fires at the same time. For example, F6 could be directed to always prefer requests from C2 over requests from C1, or to seek resolution of the conflict from the centers, or F6 could be allowed to decide for itself which fire to fight.

Employing a Higher-Level Management Role—If C2 perceives that the long-term task volume is in line with or exceeds its local processing capability and organizational expectations (but rarely do C1 and C2 require three or more fire brigades simultaneously), C1 and C2 can share their resources more efficiently by agreeing to enable a new high-level super-manager role, C^m, to coordinate decision making of C1 and C2. This new role can be performed by either C1 or C2. Assume C1 undertakes this new role, C1^m. When either C1^s or C2^s (the original C1 and C2 roles are replaced with "subordinate" center roles that have diminished responsibilities due to the enabling of the C^m role) is overloaded with fires to extinguish, C1^m requests (as part of its organizational responsibilities) status information of C2^s's fire brigades, makes a more coordinated allocation decision, and notifies C2^s to execute the part of the decision that requires its fire brigades.

3 An Extended BDI Architecture for an OAA

Our OAA architecture is built on a variant of the classic BDI model [13]. To perform the decision-making required by an OAA, we introduce organizational-level reasoning that affects the normal operational-level reasoning by incorporating organizational guidelines as beliefs and modifying the goal-evaluation mechanism used for deciding which operational goals to pursue at any moment. In addition to problem-domain beliefs, goals, and plans, we explicitly represent beliefs, goals, and plans that support organizational-level reasoning. Conventional BDI agents make decisions based on their local interests. For an OAA, however, organizational guidelines and requests from other agents also affect the agent's operational decisions. To enable the OAA to balance the potentially conflicting interests of organization, others, and self, we structure the evaluation function of operational goals to include three weighted components that explicitly represent these three interests. We will discuss these extensions after we briefly describe the basic BDI architecture.

A BDI agent has a *belief set*, a set of *primitive actions*, and a *goal library*. Its belief set is composed of *beliefs* that represent the agent's knowledge about the world, including its local state, the surrounding environment, and in an OAA, the organization, and the interests of other agents. A belief is defined by a set of variables and associated values. Primitive actions are executable, which provide the capabilities of the agent. The goal library contains a set of generic goals. A generic goal can be instantiated when its precondition is satisfied. A generic goal is specified by the following components:

Type—The label that indicates the type of the goal.

Precondition—A logical condition that must be true in order to instantiate the goal. The condition is verified on the agent's belief set. This condition does not have to remain true during the execution of the goal's plan.

Postcondition—A logical condition whose value becomes true once the goal is achieved. This condition is also verified on the agent's belief set.

Plan algorithm—The method that generates a plan to achieve the goal based on the belief set.

Utility function—The function that computes the utility of the goal given the belief set.

Algorithm 1: The BDI-Agent Reasoning Process

```
   input: Goal library $GL$ and initial belief set $B_0$
1  $B \leftarrow B_0$;        /* $B_0$ is the initial belief set*/
2  $cg \leftarrow nil$;        /* Initialize with no committed goal */
3  repeat
4      Get percept $p$;
5      $B \leftarrow updateBeliefSet(B, p)$;
6      $G \leftarrow options(B, GL, cg)$;        /* Instantiate top-level goals */
7      $cg \leftarrow filter(B, G, cg)$;        /* Select and commit a goal from goal set $G$ */
8      $\pi \leftarrow plan(B, cg)$;
9      $executePlan(B, GL, cg, cg, \pi)$;        /* recursively execute the plan */
10 until the process is terminated
```

Algorithm 1 shows the reasoning process of a BDI agent. When the agent perceives the environment, receives messages from other agents, or notes a change in the status of one of its ongoing actions, it updates its belief set. Based on the updated belief set, the agent instantiates and evaluates goals (which are also called *desires*). The agent then determines the goal(s) with the highest utility and commits to achieving them. A committed goal is also called an *intention*. Once committed to a goal, the agent will select and instantiate a *plan* from its plan library (or generate a plan by using a predefined algorithm) in order to achieve the goal. A plan for a goal consists of a sequence of primitive actions or subgoals. Goals and plans form a hierarchy, called a *goal-plan tree*. Note that committed goals only refer to top-level goals in the hierarchy.

Algorithm 2 illustrates how an agent executes a plan to achieve a top-level committed goal. The execution of a plan is generally sequential. But if a plan contains subgoals, it will be executed recursively. During plan execution, the agent will constantly update its belief set based on its perceptions and potentially instantiate new top-level goals. If a newly instantiated goal has a higher utility than a currently committed goal (and the agent cannot execute plans for them both), the method $reconsider(B, cg, G)$ will return true and the execution of the plan of that currently committed goal will then exit. Without considering the uncertainty of the environment, the satisfaction of a top-level goal requires the satisfaction of all subgoals. If the environment is uncertain or an agent has only partial observability, it is possible that the goal is achieved without achieving all subgoals of its plan or the goal is not achieved even when all subgoals of its plan are achieved. Given uncertainty of the environment and of the effects of the actions, the agent needs to check at every time step whether the goal is achieved and, if not, that its plan remains sound. This verification can be done by checking if the precondition of the first primitive action or subgoal of the plan is satisfied. If it is not, the agent needs to create a new plan for the goal.

An OAA reasons at two different levels: organizational and operational. At the organizational level, the agent reasons about whether current organizational guidelines are effective and, if not, how it can improve them by either performing a local organization adaptation or requesting a non-local adaptation from the organization designer. At the operational level, the agent decides which operational goals to pursue and which plans to choose for achieving committed goals. These two types of reasoning focus on goals at different temporal scales.

```
Algorithm 2: Function $executePlan(B, GL, cg, g, \pi)$
input  : Belief $B$, Goal library $GL$, committed goal $cg$, current goal $g$, plan $\pi_g$
output: return $false$ if committed goal $cg$ needs to be reconsidered; otherwise,
        return $true$
while not (empty($\pi$) or succeeded($g, B$) or impossible($g, B$)) do
    $\alpha \leftarrow first(\pi)$;
    $\pi \leftarrow rest(\pi)$;
    if isPrimitiveAction($\alpha$) then
        do($\alpha$);
    else
        $\pi_\alpha \leftarrow plan(B, \alpha)$;
        if executePlan(B, GL, cg, $\alpha, \pi_\alpha$) then
            return false;        /* Committed goal $cg$ needs to be reconsidered */
        end
    end
    Get percept $p$;
    $B \leftarrow updateBeliefSet(B, p)$;
    $G \leftarrow options(B, GL, cg)$;        /* Instantiate top-level goals */
    if reconsider(B, cg, G) then
        return false;
    end
    if not sound($\pi, B, g$) then
        $\pi \leftarrow replan(B, g)$;
    end
end
return true;
```

Organizational-level reasoning is intended to achieve long-term organizational goals (organizational objectives), while operational-level reasoning focuses on immediate operational goals. However, these two reasoning levels interact. Organizational guidelines (reasoned about at the organizational level) provide preferences or constraints for an OAA's choice of operational goals and plans. These guidelines specify what roles the agent should perform, priorities for these roles, and how these roles are parametrized (e.g., under what conditions these roles should be activated, who the agent should communicate with, etc.). In turn, the effectiveness of organizational guidelines is reflected in how well operational goals (weighted by their importance) are accomplished by the agent. Thus, OAA reasoning needs to include beliefs about the status of operational goals.

The top-level goal of organizational-level reasoning is to assess and maintain the effectiveness of an agent's organizational guidelines. Each OAA has three generic plans to achieve this goal that can be invoked in different situations: using existing organizational guidelines, performing localized organizational adaptation, and requesting non-local adaptation from the organizational designer. For example, if current guidelines perform as expected, the agent needs to follow them for making decisions at the operational level; otherwise, it needs to decide whether to adapt locally or more globally. Its decision-making relies on its knowledge about the current situation, i.e., its belief set. At the organizational level, the agent perceives messages from the organizational designer or other agents that perform organizational adaptation. Its belief set contains current organizational guidelines, performance annotations, and environmental expectations. To evaluate the effectiveness of the current organizational guidelines, the OAA needs to monitor the results of its operational goals over time, updating beliefs about its own performance (e.g., the failure frequency of operational goals). In addition, its belief set also needs to reflect environment changes to determine whether the changes are within its expectations, a transient deviation from its expectations, a frequent, long-duration deviation from its expectations, or a permanent deviation from its expectations [8].

At the operational level, the agent's perceptions, belief set, and goals are usually domain-specific, which is similar to the implementation of conventional BDI agent architectures. However, as agents interact and perform within an organization, the decision-making of each OAA needs to balance the interests of itself, the organization, and other agents. As we discussed above, in our OAA architecture, the goal evaluation function largely determines the agent's decision-making process. Therefore, we structure the utility function of goals to reflect explicit separation of organizational, social, and self-interests. In our current OAA implementation, the utility function $U(g, B)$ for operational goal g with belief set B is defined as the following:

$$U(g, B) = v(g)(w_s p_s(g, B) + w_o p_o(g, B) + w_r p_r(g, B)),$$

where $v(g)$ is the objective value of goal g, $p_s(g, B)$, $p_o(g, B)$, and $p_r(g, B)$ are the preferences for goal g of the agent itself, the organization, and other agents, whose values are in the range $[-1, 1]$, and w_s, w_o, and w_r are weights, whose values are in the range $[0, 1]$ and are normalized to sum to 1. These weights are also part of an agent's beliefs. An agent can change these weights through updating its belief set based on its assessment of the effectiveness of organizational guidelines, its own experiences, and requests of other agents. The local preference $p_s(g, B)$ reflects the agent's local interest for this goal. The preference $p_o(g, B)$ is determined using organizational guidelines contained in the agent's belief set B. The value $p_r(g, B)$ summarizes the agent's belief of the interest that other agents have expressed (via sent requests) for goal g. Let $p_{r1}(g, B), \ldots, p_{rk}(g, B)$ be the interests of other agents for goal g. In our current implementation, the preference $p_r(g, B)$ is defined as the following:

$$p_r(g, B) = \frac{2}{1 + \alpha e^{(-\beta \sum_{i=1}^{k} p_{ri}(g,B))}} - 1,$$

where $\alpha > 0$ and $\beta > 0$ are constants (i.e., $\alpha = 50$ and $\beta = 8$). It is possible that, for some goal, there are no organizational guidelines relevant to it or no requests from other agents. If this happens, we set the corresponding weights to zero and normalize the weights so that they sum to 1.

4 Evaluation

We are only beginning development of a complete OAA architecture that provides the complete adaptive capabilities described in Section 2. However, we believe that the basic ability to incorporate and balance organizational, social,

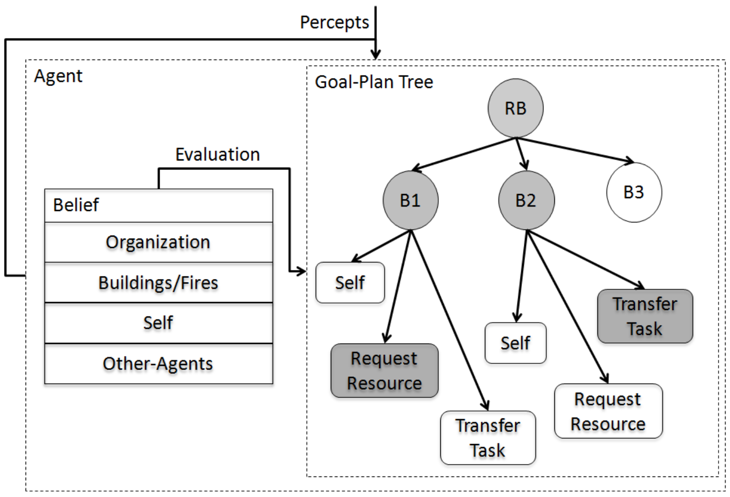


Fig. 4. Center OAA architecture (with an example goal-plan tree). Shown are a general Rescue Buildings (RB) goal and three goals specific to buildings on fire (B1, B2, and B3). Three plans are shown: extinguish building using the center's own resources, request resources from neighboring centers, and transfer the extinguish task to neighboring centers. Selected plans and goals are highlighted.

and self-interests in making operational decisions is a crucial requirement for OAAs that can perform reasonably when there is uncertainty and inappropriateness from the perspective of a single interest. Furthermore, the ability for an OAA to adjust the influence it grants each of these perspectives over time is important to agent and organizational robustness and effectiveness. For example, an OAA must be able to function without organizational guidelines (by increasing the weight of self and social influences) if no guidelines have been supplied or if they become inappropriate. In the next section, we discuss how we used the OAA reasoning architecture as the operational decision engine for agents extinguishing fires in RoboCup Rescue.

4.1 Using the OAA Architecture in RoboCup Rescue

We implemented the OAA architecture for centers in RoboCup Rescue (Fig. 4) Each agent receives percepts, from itself (characterizing the internal status of the agent), from external factors in the environment (from the RoboCup Rescue simulator), and from other agents. These percepts include organizational guidelines received from an automated designer[5] (e.g. the geographic region the center is managing), tasks (e.g. fires in the region, task-transfer requests), and resource statuses (e.g. water levels of fire-brigades the center is controlling, resource-transfer requests). Based on these percepts, each agent updates the following types of beliefs:

[5] Not discussed in this paper.

Organization Beliefs (OB)—represent the organizational guidelines given an agent including the roles it is responsible for and specific parameters of those roles. For example, a role that a center performs is task allocator. The guidelines parametrize that role with the fire-brigade agents that the center has authority over and the geographical region it is responsible for managing.

Building/Fire Beliefs (BB)—represent the various buildings in the environment, whether they are on fire or not, the importance of the buildings, and the utility received on extinguishing fires in those buildings.

Resource Beliefs (RB)—status information regarding fire-brigades agents directed by the center, including their water level, health, and location.

Other Center Beliefs (CB)—assessment of the willingness of other centers to help this center and of the likelihood of other centers to request help from this center based on past experiences.

As beliefs are updated, each agent forms a local goal-plan tree, where each goal and plan is evaluated based the agent's beliefs. Each center starts with one inherent goal, to extinguish buildings that are on fire. As a new fire percept is received, the agent instantiates a subgoal (B1 for example) for the particular building. For each building on fire, the agent can execute one of three plans: 1) it could try to extinguish the fire using only its own brigades (Self in Fig. 4), 2) it could augment its own resources by requesting brigades from neighboring centers (Request Resources), or 3) it could transfer the task of extinguishing the fire to a neighboring center (Transfer Task). The utility of a goal is calculated using the following function:

$$\begin{aligned} U(goal, beliefset) = f(goal, BB)(w_s f_s(goal, RB) \\ + w_o f_o(goal, OB) + w_r f_r(goal, CB)) \end{aligned}$$

Thus, if a center is highly inclined to follow its organizational guidelines, its w_o will be high (e.g., 0.8), while w_s and w_r will be low (e.g., 0.1 each).[6] If a center is strongly self-motivated, w_s will be high. If a center greatly prefers assisting other agents, w_r will be high. These weights are normalized so that $w_s + w_o + w_r = 1$. Note that there are two things operating here. For each goal, its objective utility ($f(goal, BB)$) is boosted by how much the goal agrees with the center's organizational guidelines ($f_O(goal, OB)$), its beliefs about itself ($f_S(goal, RB)$), and its beliefs about its neighboring agents ($f_R(goal, CB)$). On the other hand, the weights influence the center's inclination for goals that agree with its organization-centric, social-centric, and self-motivated interests.

4.2 Demonstrations

We illustrate OAA behavior differences given different (fixed) weights for w_s, w_o, and w_r using RoboCup Rescue scenarios. Consider center C2 and two buildings B1 and B2, both on fire (Fig. 5).

[6] Although we wanted to show the different behavior of OAAs that were heavily weighted toward one of the three interest perspectives, we did not want to eliminate entirely the ability to choose activities stemming from the other two perspectives.

Initially, $w_s = w_o = w_r = \frac{1}{3}$. When the fires are discovered, C2 generates a goal to extinguish each fire (goals B1 and B2). For goal B1, $f_S = \frac{1}{d}$, $f_O = 1$ (since building B1 is within the region of C2's organizational responsibility), and $f_R = 0$ (since extinguishing building B1 has not been requested of C2 by another agent). For goal B2, $f_S = \frac{1}{d}$, $f_O = 0$ (since building B2 is outside the responsibility region that is specified in C2's organizational guidelines), and $f_R = 0$. Because of the added organizational inclination, C2 will prefer accomplishing goal B1 over B2, unless B2 is a significantly more important building than B1 (i.e $f_{B1} << f_{B2}$). Once C2 assigns resources to accomplish goal B1, and if $U(B2, B)$ is greater than a certain threshold, C2 will generate plans for requesting help from neighboring centers for accomplishing B2. Values for f_s, f_o, f_r will be calculated for the goals associated with each of the plans, allowing for C2 to select one of them.

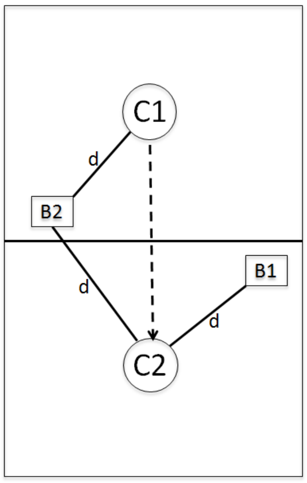


Fig. 5. Center C2 discovers two buildings on fire, B1 and B2 at an equal distance, d, from the center of the region. It has enough resources to handle only one of them. Center C1 discovers fire B2 and sends a request to C2 for help extinguishing it

Extending this scenario further, suppose C1 discovers B2. It will generate a corresponding goal for it and assign values $f_S = \frac{1}{d}$, $f_O = 1$, and $f_R = 0$. Assume C1 does not have enough resources to fight fire B2, and from its perspective $U(B2, B)$ is greater than its threshold. C1 will generate plans for requesting help from neighboring agents (just as C2 did previously). Suppose C1 makes a request of C2 to extinguish this fire. This means C2 will increase its f_R value for goal B2 based on the local importance of handling requests from C1. Now, C2 must make an operational decision between accomplishing goals B1 and B2. If rather than the weights being $\frac{1}{3}$ each, $w_r > w_o$, C2 will assign its resources towards accomplishing goal B2. If $w_r < w_o$, C2 will assign its resources to B1. If both C1 and C2 were to favor self-interests, they both might try to request resources for B2 (because the organizational interest is not accounted for), incurring both the operational cost of making those duplicate requests, as well as the potential cost of over-assigning resources to B2.

These initial demonstrations show that our RoboCup Rescue OAAs not only behave differently when strongly weighted toward different interests (organization, social, self), but that they perform ineffectively when the interest they are emphasizing does not fit their beliefs and environment. More importantly, these demonstrations show the potential for OAAs that maintain a balance between these three important interests and adjust that balance when long-term environmental conditions and agent behaviors warrant. Our OAAs can follow organizational guidelines when they are available, yet operate in their absence or ignore them if they are inappropriate.

Acknowledgement. This material is based in part upon work supported by the National Science Foundation under Award No. IIS-0964590. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. C. Bernon, V. Chevrier, V. Hilaire, and P. Marrow. Applications of self-organising multi-agent systems: An initial framework for comparison. *Informatica*, 30:73–82, 2006.
2. E. Bonabeau, M. Dorigo, and G. Théraulaz. *Swarm Intelligence: From natural to artificial systems.* Oxford University Press, 1999.
3. C. H. Brooks and E. H. Durfee. Congregation formation in multiagent systems. *Journal of Autonomous Agents and Multiagent Systems*, 7:145–170, 2003.
4. D. D. Corkill. *A Framework for Organizational Self-Design in Distributed Problem-Solving Networks.* PhD thesis, University of Massachusetts Amherst, Feb. 1983.
5. D. D. Corkill and S. E. Lander. Agent organizations. *Object Magazine*, 8(4):41–47, Apr. 1998.
6. E. H. Durfee and Y. pa So. The effects of runtime coordination strategies within static organizations. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 612–618, Nagoya, Japan, Aug. 1997.
7. L. Gasser and T. Ishida. A dynamic organizational architecture for adaptive problem solving. In *Proceedings of the National Conference on Artificial Intelligence*, pages 185–190, Anaheim, California, July 1991.
8. B. Horling, B. Benyo, and V. Lesser. Using self-diagnosis to adapt organizational structures. In *Proceedings of the Fifth International Conference on Autonomous Agents*, pages 529–536, Montreal, Canada, June 2001.
9. B. Horling and V. Lesser. Using quantitative models to search for appropriate organizational designs. *Autonomous Agents and Multi-Agent Systems*, 16(2):95–149, 2008.
10. S. Kamboj and K. S. Decker. Organizational self-design in semi-dynamic environments. In *Proceedings of the 2007 IJCAI workshop on Agent Organizations: Models and Simulations* (AOMS-07), pages 335–337, Jan. 2007.
11. H. Kitano and S. Tadokoro. RoboCup-Rescue: A grand challenge for multi-agent and intelligent systems. *AI Magazine*, 22(1):39–52, 2001.
12. J. G. March and H. A. Simon. *Organizations.* John Wiley & Sons, 1958.
13. A. S. Rao and M. P. Georgeff. Bdi agents: From theory to practice. In *Proceedings of the First International Conference on Multi-Agent Systems* (ICMAS'95), pages 312–319, San Francisco, California, June 1995.
14. G. D. M. Serugendo, M.-P. Gleizes, and A. Karageorgos. Self-organisation and emergence in MAS: An overview. *Informatica*, 30:1–11, 2006.
15. M. Sims, D. Corkill, and V. Lesser. Automated organization design for multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 16(2):151–185, Apr. 2008.
16. T. Sugawara and V. Lesser. Learning to improve coordinated actions in cooperative distributed problem-solving environments. *Machine Learning*, 33(2-3):129–153, Nov. 1998.
17. M. Tambe, J. Adibi, Y. Alonaizon, A. Erdem, G. Kaminka, S. Marsella, and I. Muslea. Building agent teams using an explicit teamwork model and learning. *Artificial Intelligence*, 110:215–240, 1999.