An Organizational Model for Designing Adaptive Multiagent Systems

Scott A. DeLoach **Eric Matson**

Multiagent and Cooperative Robotics Laboratory
Department of Computing and Information Sciences, Kansas State University
234 Nichols Hall, Manhattan, Kansas 66506, USA
{sdeloach, matson}@cis.ksu.edu

Abstract

This paper describes how to design adaptive multiagent systems using an *organizational model*, which defines the entities and relationships of a typical organization. The major elements of the model consist of goals, roles, agents, capabilities, and the relationships between them. By designing a system using the model, the system can determine the best mapping of agents to roles, based on their current capabilities, for the current system goals. Specifically, we show how to use the model in developing an adaptive information system, which is an information system that can modify its processing algorithms or information sources to provide required information at various levels of efficiency and effectiveness.

Introduction

The field of multiagent systems has long been interested in using social and biological concepts to help create architectures for adaptive systems architectures. Social structures, specifically *organizations*, provide a framework for analyzing the roles various individuals play and how they interact to achieve organization-wide goals. This framework is closely related to the overall goal of multiagent systems, which is to build intelligent systems that can adapt effectively to changes internally within the system or externally within their environment. The organizational framework also provides a method for describing the relationships between organizational goals and individual goals.

Our approach to using organizational concepts in building multiagent systems is to provide an organizational framework in the form of an organizational model from which organization-based multiagent systems can be developed. Our model defines the set of entities required in an organization as well as the valid relationships that exist between those entities.

The organizational framework presented here was derived from the underlying object model associated with the Multiagent Systems Engineering methodology and its associated development tool, agentTool (DeLoach & Wood 2001). In MaSE, goals are derived from the initial description, which are then assigned to a set of roles to achieve. Finally, agent classes are designed that can actually play the required roles. Thus, the organization model captures the notions central to agent-oriented software engineering, namely goals, roles, and agents.

In the remainder of this paper, we shall first discuss other work related to building multiagent teams. Then, we briefly discuss the problem that we will use as an example of how to use our organizational model, that of an adaptive information system. Next we present our organizational model followed by an example of using the model to develop an adaptive information system. We end with a discussion of our current conclusions and future work.

Related Work

Computational organization theory uses mathematical and computational techniques to study both human and artificial organizations (Carley 1998). While organizational concepts are not exclusive to computational organization theory, results from the field are illuminating. Specifically, they suggest that organizations tend to adapt to increase performance or efficiency, that "the most successful organizations tend to be highly flexible" (Carley 1998), and that the best organizational designs are highly application and situation dependent (Lawrence and Lorsch 1967). It also provides findings about the conditions under which certain organizations work best. For instance, as the number of hierarchical levels in an organization increases, efficiency and effectiveness tends to decrease while decentralized organizations tend to have higher performance. However, hierarchical organizations tend to exhibit higher reliability (Carley 1995). These insights seem to suggest that allowing teams to determine their organization at runtime, as we propose, could have positive effects on team performance.

There have been several attempts at formalizing the concepts of teamwork within an organization in the area of multiagent systems. While efforts such as Joint Intentions (Cohen and Levesque 1990, 1991) (Jennings 1993, 1995), Shared Plans (Grosz and Kraus 1996), and Planned Team

Activity (Kinny et. al. 1992), have been proposed and even implemented (Tambe 1997), they fail to provide straightforward and easily adaptable concepts for wide spread development of such systems.

Other closely related work includes the CoDA project at the University of Maine (Turner and Turner 2001). The CoDA project deals with a team of autonomous underwater vehicles that must self-organize and reorganize using a two level strategy where a meta-level organization designs a task-level organization to carry out team goals.

Problem

Our goal is to develop an adaptive information system (AIS) based on our organizational model. An AIS is an information system that can modify its processing algorithms or information sources to provide required information at various levels of efficiency and effectiveness. In general, an AIS selects the best available data and fuses it in an attempt to answer queries from AIS users. For this problem, we pursue a simple example, which might be only part of a typical AIS. Specifically, we assume we are interested in answering only one possible query from a battlefield commander: *produce a list of the enemies moving vehicles within a given area*. To answer this query, the AIS must be able to use appropriate sensors to provide information about moving vehicles and vehicle identification. In our example, we have three types of information sources at our disposal: moving vehicles sensors, vehicle identification sensors, and a database of enemy/friendly vehicle types. Ideally, the AIS will take information from one or more of information sources to produce the required information for a specific *area of interest*. Here we assume the only reason to use multiple sensors is that their area of coverage does not cover the entire area of interest. However, if a single sensor does cover that area, we will use the single sensor to provide the required information. In general, the information fusion process follows the flow shown in Figure 1.

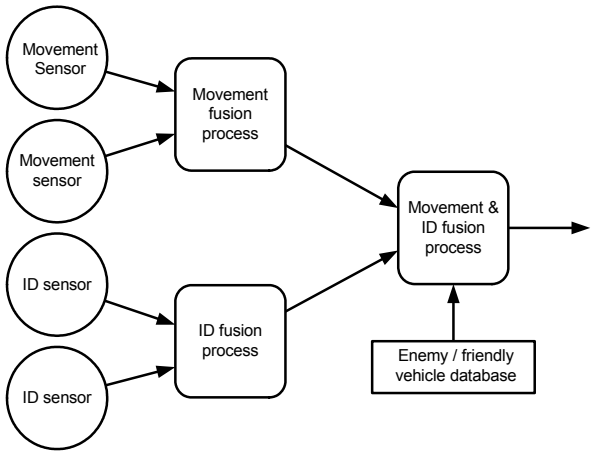

Figure 1. AIS Information Flow

Organization Model

To implement teams of autonomous, heterogeneous agents, we created an *organizational model*, which defines and constrains the required elements of a stable, adaptable and versatile team. While most people have an intuitive idea of what an organization is, there is not a universal definition. However, in most organizational research, organizations have typically been understood as including *agents* playing *roles* within a *structure* in order to satisfy a given set of *goals*. Our proposed *organizational model* (O) contains a structural model, a state model and a transition function.

$$O = < O_{struct}, O_{state}, O_{trans}>$$

Figure 2 shows the combined structural and state models using standard UML notation. The *structural model* includes a set of goals (*G*) that the team is attempting to achieve, a set of roles (*R*) that must be played to attain those goals, a set of capabilities (*C*) required to play those roles, and a set of rules or laws (*L*) that constrain the organization. The model also contains static relations between roles and goals (*achieves*), roles and capabilities (*requires*), individual roles (*related*), and goals (*subgoal* and *precedes*). Formally, we model the organization structure as a tuple.

$$O_{struct} = <G, R, L, C, achieves, related, requires, subgoal, precedes>$$

where

$$achieves: R, G \rightarrow 0 .. 1$$
$$related: R, R \rightarrow Boolean$$
$$requires: R, C \rightarrow Boolean$$
$$subgoal: G, G \rightarrow Boolean$$
$$precedes: G, G \rightarrow Boolean$$

The team *goals* include the goal definitions, goal-subgoal decomposition, and the relationship between the

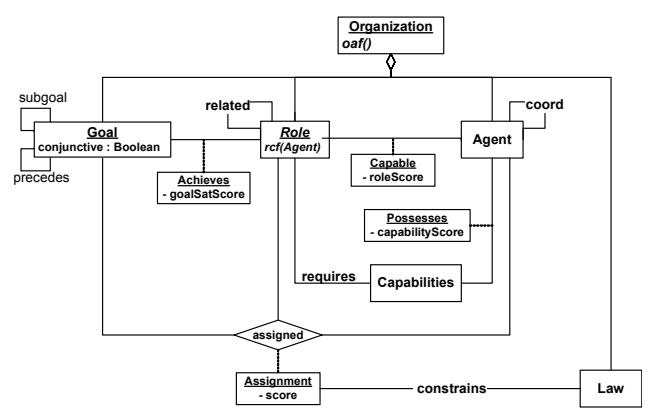

Figure 2. Organization Model

goals and their subgoals, which are either conjunctive or disjunctive.The *subgoal* predicate defines whether a goal, g1, is the parent goal of the second goal, g2. In general, a subgoal relationship is 1 to many (0 or more). We can define the children of a goal as

```
children(g) = {g1:G | subgoal(g,g1)}
```

The children of a goal may be either conjunctive or disjunctive, which is denoted by the `conjunctive` predicate attached to each goal. If a goal is a conjunctive goal (g.conjunctive = true), then that goal may only be satisfied if each goal in children(g) are satisfied. Conversely, if a goal is disjunctive (g.conjunctive = false), then that goal is satisfied when any goal in children(g) is satisfied. If a goal does not have any subgoals (children(g) = {}), then the goal is a leaf node and is neither conjunctive or disjunctive. There is also a time-based relationship that exists between goals. We say goal, g1, *precedes* goal, g2, if g1 must be satisfied before any part of g2 can be satisfied. This allows the organization to work on one part of the goal tree at a time. If g1 precedes g2 (precedes(g1,g2) = true), then the organization can put its full effort into satisfying g1 without worrying about g2 until it has achieved g1.

Roles define parts or positions that an agent may play in the team organization. In general, roles may be played by zero, one, or many agents simultaneously while agents may also play many roles at the same time. Each role requires a set of *capabilities*, which are inherent to particular agents and may include data access capabilities, data manipulation capabilities, or computational capabilities. Typically, an agent's capabilities are dynamic; they may improve or degrade over time, often causing team reorganization.

Organizational rules are used to constrain the assignment of agents to roles and goals within the organization. Generic rules such as "an agent may only play one role at a time" or "agents may only work on a single goal at a time" are common. However, rules are often application specific, such as requiring particular agents to play specific roles.

The structural model relations define mappings between the structural model components described above. A role that can be used to satisfy a particular goal is said to *achieve* that goal, while a role *requires* specific capabilities and may work directly with other roles, thus being *related* to those roles. *Achieves* is modeled as a function to capture the relative ability of roles to satisfy a given goal.

The *organizational state model* defines an instance of a team's organization and includes a set of agents (*A*) and the actual relationships between the agents and the various structural model components.

$$O_{state} = \langle A, possesses, capable, assigned, coord \rangle$$

where

```
possesses: A, C → 0 .. 1
capable: A, R → 0 .. 1
assigned: A, R, G → 0 .. 1
coord: A, A → Boolean
```

An agent that *possesses* the required capabilities for a particular role is said to be *capable* of playing that role. Since not all agents are created equally, *possesses* is modeled as a real valued function, where 0 would represent absolutely no capability to play a role while a 1 indicates an excellent capability. In addition, since agent capabilities may degrade over time, this value may actually change during team operation. The *capable* function defines the ability of an agent to play a particular role and is computed based on the capabilities required to play that role. During the organization process, a specific agent is selected to play a particular role in order to satisfy a specific goal (however, this does not limit agents from playing multiple roles). This relationship is captured by the *assigned* function, which includes a real valued score that captures how well an agent, playing a specific role, can satisfy a given goal. When an agent is actually working directly with another agent, it is coordinating (*coord*) with that agent. Thus, the state model defines the current configuration for a particular organization within the structure provided by the structural model.

The *organization transition function* defines how the organization may transition from one organizational state to another over the lifetime of the organization, $O_{state(n)} \rightarrow O_{state(n+1)}$. Since the team members (agents) as well as their individual capabilities may change over time, this function cannot be predefined, but must be computed based on the current state, the goals that are still being pursued, and the organizational rules. In our present research with purely autonomous teams, we have only considered reorganization that involves the *state* of the organization. However, we have defined two distinct types or reorganization: *state reorganization*, which only allows the modification of the organization state, and *structure reorganization*, which allows modification of the organization structure (and may require state reorganization to keep the organization consistent). To define state reorganization, we simply need to impose the restriction that

$$O_{trans}(O).O_{struct} = O.O_{struct}$$

Technically, this restriction only allows changes to the set of agents, *A*, the *coord* relation, and the *possesses*, *capable*, and *assigned* functions. However, not all these components are actually under the control of the organization. For our purposes, we assume that agents may enter or leave organizations or relationships, but that these actions are triggers that cause reorganizations and are

not the result of reorganizations. Likewise, *possesses* (and thus *capable* as well) is an automatic calculation on the part of an agent that determines the roles that it can play in the organization. This calculation is totally under control of the agent (i.e. the agent may lie) and the organization can only use this information in deciding its organizational structure. Changes in an agent's capabilities may also trigger reorganization. That leaves the two elements that can be modified via state reorganization: *assigned* and *coord*. Thus, we define state reorganization as:

$$O_{trans}(state) : O \rightarrow O$$

where

$$O_{trans}(state)(O).O_{struct} = O.O_{struct}$$
$$\wedge\ O_{trans}(state)(O).O_{state}.A = O_{state}.A$$
$$\wedge\ O_{trans}(state)(O).O_{state}.possesses = O_{state}.possesses$$
$$\wedge\ O_{trans}(state)(O).O_{state}.capable = O_{state}.capable$$

Validity and Viability Constraints

This section describes the concepts of valid and viable organizations. In general, a *valid organization* is one in which the structure satisfies all the constraints of the organization model. A *viable organization*, on the other hand, is one in which the appropriate agents and roles exist such that an assignments of agents to roles to goals exists that can reasonably be expected to allow the team to reach its overall team goals.

Validity. As described above, to be valid, an organization must have a structure that would allow it to form an organization, given the right collections of goals, roles, and agents. Thus, these constraints are further restrictions on the model presented above. First, we deal with the capabilities required by roles and agents. Since the objective of the model is to define teams of agents that can work together, it only makes sense that agents have some capability, even if it is purely computational or communicative. Likewise, each role must have some basic requirements as well. Therefore, we require that all roles *require* at least one capability and that all agents *possess* at least one capability, the later being denoted by a capability score greater than zero (# is the cardinality of the set).

$$\forall\ r{:}R \quad \#\{c \mid requires(r,c)\} \geq 1 \qquad \text{(C1)}$$

$$\forall\ a{:}A \quad \#\{c \mid possesses(a,c) > 0\} \geq 1 \qquad \text{(C2)}$$

The next set of constraints deal with capabilities of agents and roles as they relate to assignments. To be *capable* of playing a role in the current organization, an agent must *possess* all the capabilities that are *required* of that role. It also follows that an agent must be *capable* of playing a role before it can be *assigned* to play that role.

$$\forall\ a{:}A,\ r{:}R \quad capable(a,r) > 0 \Leftrightarrow \{c \mid requires(r,c)\} \subseteq \{c \mid possesses(a,c)\} \qquad \text{(C3)}$$

$$\forall a{:}A,\ r{:}R,\ g{:}G \quad (assigned(a,\ r,\ g) > 0) \Rightarrow capable(a,r) > 0 \qquad \text{(C4)}$$

In order to allow the preceding constraints to be true, it is necessary that the set of capabilities in an organization (C) include all the capabilities required by all roles within the current organization.

$$\forall\ r{:}R \mid \{c \mid requires(r,c)\} \subseteq C \qquad \text{(C5)}$$

The next constraint concerns the relationship between the *related* relationship between roles and the *coord* relationship between agents. Because roles define the part the agents will be playing and the *related* relationship defines valid relationships between those roles, agents should only have coordination relationships (coord) if they are playing the appropriate roles.

$$\forall\ a1,\ a2{:}A \quad coord(a1,\ a2) \Rightarrow (\exists\ r1,\ r2{:}R,\ g1,\ g2{:}G\ ((assigned(a1,r1,g1) > 0) \wedge (assigned(a2,r2,g2) > 0) \wedge related(r1,\ r2))) \qquad \text{(C6)}$$

The final validity constraints are straightforward structural constraints that require the related and coord relations to be symmetric. That is, if one role is related to another role, then the second role must also be related to the first role.

$$\forall\ r,\ r1{:}R\ related(r,r1) \Rightarrow related(r1,r) \qquad \text{(C7)}$$

$$\forall\ a,\ a1{:}A\ coord(a,\ a1) \Rightarrow coord(a1,\ a) \qquad \text{(C8)}$$

Viability. Even though an organization may be valid (using the constraints above), it does not mean that there will actually be an instance of the organization that can actually satisfy the goals of the organization. In order to satisfy the organizational goals, the team must have the right mix of roles to satisfy the goals and agents to play the required roles. The viability constraints presented below are only the base viability constraints. Often, viability constraints are application specific and are embedded in the organization in the form of organizational rules.

The first viability rule ensures that there is some real organization that actually exists to solve some goal. Thus we require that there must be at lest one element each of goals, roles, and agents to have a viable organization. However, since this requirement follows for roles and agents given the definition of viability below (which requires an that some role be available to achieve each goal and an agent that is capable of playing that role) we can simply require the organization to have at least one goal.

$$\#\ G \geq 1 \qquad \text{(C9)}$$

For an organization to be truly viable, one would expect it to be able to achieve its overall goal. Therefore, we define a *viable* organization as an organization that is able to show that the overall organization goal is achievable by some set of assignments of goals, roles, and agents. Thus given some overall goal, g_o, we can define organization viability as

$$\texttt{satisfiable}(g_o) \qquad \text{(C10)}$$

where the *satisfiable* predicate as defined below.

```
satisfiable(g) =
 children(g) = {} ⇒
    (∃ a:A r:R
          achieves(r, g) ∧ capable(a, r) > 0))
 children(g) != {} ⇒
    (g.conjunctive
         ⇒ ∀ g':children(g) satisfiable(g')
     ¬g.conjunctive
         ⇒ ∃ g':children(g) satisfiable(g'))
```

Notice that this does not require all goals be satisfied (since we allow the notion of disjunctive goals in which only one subgoal must be satisfied) nor does it ensure that there are enough of the right types of agents to achieve the organization's goal. It merely states that it might be possible to find a suitable set of assignments so that the overall goals of the organization may be achieved.

Capability

So far, we have used the term capability generically. However, we need to define it more precisely before preceding. A capability's existence is based on the collective sense in which it is viewed. To specify this we further define capabilities in relation to agent and roles that exist within a self-reorganizing multiagent team. As described above, an agent *possesses* specific capabilities while roles *require* particular capabilities, each with specific scores.

The *capability set of an agent*, C_a, varies from the empty set, if the agent possesses no capability, to a complete set of the capabilities that the agent intrinsically possesses. Normally even a simple agent has multiple capabilities.

$$C_a(a) = \{c \mid \texttt{possesses}(a,c)\}$$

Likewise, the *capability set of a role*, C_r, is the set of capabilities required to play that specific role. All non-trivial roles must have at least one capability in order to accomplish some task or goal.

$$C_r(r) = \{c \mid \texttt{requires}(r,c)\}$$

The capability of an agent, *a*, to play a specific role, *r*, are application and role specific. To capture this concept, we have defined a role capability function, *rcf*, which is defined uniquely for each role. This allows the designer of each role to specify how specific capabilities affect the ability of an agent to play that role. Some capabilities may be extremely important, while others are not. If an agent does not possess a required capability (*possesses(a,c) = 0*), then the agent has no capacity to play that role and thus the rcf = 0. Thus, the *capability score* of an agent playing a particular role is defined as

$$\texttt{capable}(a,r) = r.\texttt{rcf}(a)$$

Organization Selection

We generally assume that an organization strives to operate using an optimal configuration. Ideally, an organization will select the best set of assignments to maximize its ability to achieve its goals, which requires maximizing its organizational capability score, *Os*. As in the case of determining the ability of agents to play specific roles, the selection of assignments is also application specific. Thus, each organization has its own organization assignment function, *oaf*, which computes the organizational capability score, Os, based on individual sets of assignments. Using the oaf, the organization developer can specify how to make assignments based on a variety of organization specific constraints such as the importance of the specific goals or whether the assignment of multiple agents to a given role and goal will improve goal satisfaction. In the absence of an organization-specific organizational assignment function, we often just sum the assignment scores as shown below.

$$\texttt{Os} = \sum_{\forall a,r,g} \texttt{assigned}(a,r,g)$$

where *assigned(a,r,g) = 0* if that agent is not assigned to play a specific role to satisfy a goal.

Reorganization Triggers

There are a variety of events that may occur in the lifecycle of a multiagent team that may require it to reorganize. In general, *reorganization* is initiated when an event occurs such that the team (1) has reached a goal or subgoal, (2) is no longer capable of reaching its overall goal, or (3) realizes that it could reach its goal in a more efficient or effective manner. When the team is no longer capable of reaching its overall goal, we call this a *goal failure*. We have currently identified three role-related goal failure scenarios:

1. When a required role has not been assigned
2. When an agent relinquishes some required role
3. When an agent suffers a failure that keeps it from accomplishing its role

When a team reorganizes for *efficiency*, it is accomplishing its goals; it is just not doing so as efficiently as possible. In an information system, we equate efficiency to timeliness. Thus, if we have a requirement to produce new

information by a set deadline, or consistently every few minutes, it may have to reorganize in order to meet those deadlines. Reorganizing for *effectiveness* can be equated to information quality. In an intelligence gathering system, this is often quantified in terms of a confidence level. In this case, a commander might need a certain confidence level in a piece of information before making a decision and timeliness may be traded for quality.

Assuming efficiency and effectiveness requirements are modeled as goals, then the *capable* function captures all the data necessary to assess organizational effectiveness and efficiency. Triggering an efficiency/effectiveness based reorganization requires roles to monitor conditions, such as those discussed above.

Organization Design

This section presents a design of an adaptive information system meeting the requirements described above. To design an appropriate organization, we must define the goals, roles, capabilities, and agent types. After defining the organization, we present an example of how the organization would select the appropriate set of agent instance assignments to achieve a specific goal.

Goal Definition

Using a conjunctive goal tree (where the arc between subgoals represents conjunctive subgoals), we have developed a goal structure for our AIS as shown in Figure 3. From Figure 3, we can extract the leaf goals that must be satisfied by roles and agents within the system.

G1 – validate enemy vehicle (1.1.1)
G2 – access moving vehicle sensor (1.1.2.1.1)
G3 – combine moving vehicles into list (1.1.2.1.2)
G4 – combine moving & ID vehicles (1.1.2.2)
G5 – access ID sensor (1.1.2.3.1)
G6 – combine vehicle IDs into list (1.1.2.3.2)

For the sake of this example, we will assume that the parameter, *interestArea*, is an attribute of the top level goal, and is available to the organization for use in determining the appropriate sensor to choose.

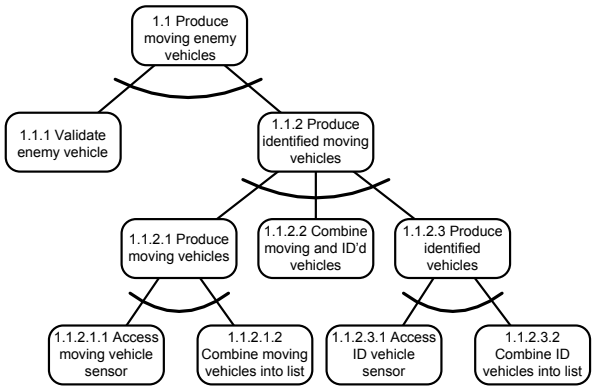

Figure 3. Partial AIS Goal Model

Role Definition

For simplicity, we map each leaf goal to a single role. In general, however, we could create multiple roles for each goal and the roles themselves could achieve multiple goals. Thus, we define the following roles in a straightforward manner:

R1 – enemy vehicle validator
R2 – moving vehicle sensor interface
R3 – moving vehicle list combiner
R4 – moving/ID combiner
R5 – ID sensor interface
R6 – ID list combiner

As described above, each role *requires* certain capabilities and has a role capability function, which describes how well an agent may play that role in light of the capability it possesses. In this domain, we define capabilities in terms of the ability to produce specific types of information for some specific area of interest. The required capabilities for each role are listed below.

R1 – enemy vehicle validator
- C1 - Capable of validation
- C2 - Access to enemy/friendly database

R2 – moving vehicle sensor interface
- C3 - Access to moving vehicle sensor
- C4 - Area of coverage

R3 – moving vehicle list combiner
- C5 - Capable of producing moving vehicles list

R4 – moving/ID combiner
- C6 - Capable of combining moving & ID lists

R5 – ID sensor interface
- C7 - Access to ID sensor
- C4 - Area of coverage

R6 – ID list combiner
- C8 - Capable of producing vehicle ID list

The *role capability function* (rcf) for each role is straightforward, with the exception of R2 and R5, the sensor interface roles. In each of the other roles, the rcf has only two values: 0 if the agent does not possess the appropriate capability and 1 if it does. However, the rcf for R2 and R5 are more complex as they must take into account the area that each sensor can coverage in relation to the area of interest. Thus the rcf for R2 and R5 are

```
R2.rcf(a)= possesses(a,C3) * possesses(a,C4)
R5.rcf(a)= possesses(a,C7) * possesses(a,C4)
```

Since `possesses(a,C3)` and `possesses(a,C7)` result in either a 0 or 1 value, multiplication with the remaining possesses value for C4 effectively selects only

agents with the appropriate information producing capability. In case of `possesses(a,C4)`, the calculation is based on the current area of interest, as defined by the value of *interestArea*. We define `possesses(a,C4)` as the percentage of the *interestArea* covered by the particular sensor attached to agent a. To ensure we get the appropriate coverage area of sensors, we include an organizational law that constrains the choice of assignments such that

$$\texttt{coverage(IDagents)} \geq \textit{interestArea} \quad \text{(L1)}$$

where

```
IDagents = {a:A | assigned(G5,R5,a) > 0}
```

and `coverage` returns the coverage area of a sensor set.

Agent Type Definition

The next step is to define the types of agents that exist within the organization and their capabilities. To play each role specified above, we define a specific type of agent for each role as listed below:

Validator (R1)
MVID_combiner (R4)
MV_combiner (R3)
ID_combiner (R6)
MV_sensor (R2)
ID_sensor (R5)

Example

To show an example of the AIS in operation, we must first populate the organization with an appropriate set of agents. Again, to simplify the example, we define only a single agent for each agent type except for the sensor type agents. Thus, our exemplar AIS has the following 14 agents.

A1: Validator
A2: MVID_combiner
A3: MV_combiner
A4: ID_combiner
A5... A9: MV_sensor
A10... A14: ID_sensor

Due to space limits, we only describe the selection process of the five ID_sensor agents. The selection of the first four agents is trivial (they are mapped one-to-one to roles), while the selection of the MV_sensor agents is identical to the selection of the ID sensors. Thus, the coverage area capabilties for the ID_sensor agents are:

A10 – coverage area = (0,1) … (3,3)
A11 – coverage area = (3,0) … (6,6)
A12 – coverage area = (0,4) … (3,8)
A13 – coverage area = (5,4) … (9,6)
A14 – coverage area = (4,7) … (9,9)

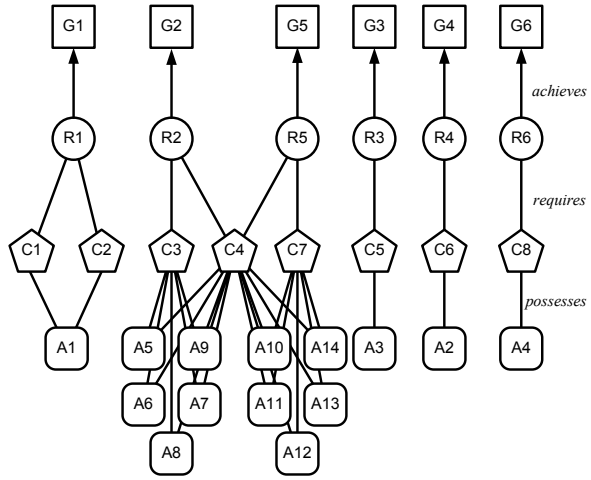

Figure 4. AIS Organization

Given the defined goals, roles, and capabilities along with our set of agents, we get the following sets in our organizational model.

G = {G1, G2, G3, G4, G5, G6}
R = {R1, R2, R3, R4, R5, R6}
A = {A1, A2, A3, A4, … , A14}
C = {C1, C2, C3, C4, C5, C6, C7, C8}

A picture of the organization is shown in Figure 4. In this organization, both the *achieves* and *requires* functions are shown as defined above. *Achieves* is a one-to-one mapping between roles (R1 ... R6) and goals (G1 ... G6) and *requires* is a mapping between roles and capabilities (C1 ... C8). The *possesses* function is shown as links between agents (A1 ... A14) to capabilities.

Given the AIS organization definition above, we now show how the organizational information is used to arrive

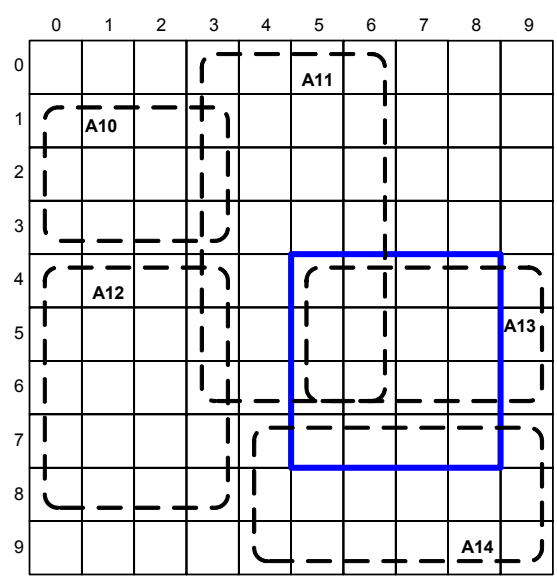

Figure 5. Grid Area

at the optimum organization for a specific query. For this example, we will assume that we are working in a 10 x 10 grid area. For our example, we assume the system receives the query to find a list of moving enemy targets within the rectangle bounded by the coordinates (4,5) and (7,8) as denoted by the rectangle with the thick border in Figure 5. From this diagram, we can see that the organization must find a way of using multiple sensor to satisfy the query goal while observing the coverage law (L1). Using the definition of *capable* and the rcf specified above for R5, we get the following *capable* values for each ID_sensor agent to play role R5.

```
A10 = 1 * 0  = 0
A11 = 1 * 6/16   = .375
A12 = 1 * 0  = 0
A13 = 1 * 12/16  = .75
A14 = 1 * 4/16   = .25
```

Using these values, we can see that any subset of {A11, A13, A14} that satisfies the coverage law (L1) will produce the required information. If we take the smallest subset that accomplishes the overall goal, we will select two agents, A13 and A14, to play R5, the ID sensor interface role.

Conclusions

In this paper, we presented an example of using our organization model to define an adaptive information system that is capable of selecting the best agents to play the appropriate roles in achieving a set of goals, which in this domain is based on user queries. Although the example presented is simplistic as does not show the full power of the approach, it does illustrate its effectiveness in selecting the appropriate assignments.

The organization model, as presented in this paper, is applicable to both multiagent systems as well as cooperative robotic applications. Besides developing an AIS simulation (Matson and DeLoach 2003), we have also developed an application that controls the use of various sensors within a single robot (Matson and DeLoach 2004).

Future enhancements to the organizational model include the inclusion of sub-organizations. The concept will allow specific roles to be played by teams of agents that are based on their own organizational model. Other related work includes developing distributed algorithms for reorganizing when goals are not being met or agents enter or leave the organization, merging of existing organizations, and learning when and how to reorganize. We also developing software engineering methods and tools for creating organization-based multiagent and cooperative robotic systems based on our Multiagent Systems Engineering (MaSE) methodology (DeLoach et. al. 2001). We plan to extend the MaSE methodology to allow a principled and straightforward development of organization based systems.

References

Carley, K. M. 1995. Computational and Mathematical Organization Theory: Perspective and Directions. *Computational and Mathematical Organization Theory* 1(1): 39 – 56.

Carley, K. M. 1998. Organizational Adaptation. *Annals of Operations Research* 75:25-47.

Cohen, P.R., and Levesque, H.J. 1990. Intention is Choice with Commitment. *Artificial Intelligence* 42(3).

Cohen, P. R. and Levesque, H. J. 1991. Teamwork. *Nous* 25(4):487-512.

DeLoach, S. A., Wood, M. F. and Sparkman, C. H., 2001. Multiagent Systems Engineering. *The International Journal of Software Engineering and Knowledge Engineering*, 11(3):231-258.

DeLoach, S. A., & Wood, M. F, Developing Multiagent Systems with agentTool. in *Intelligent Agents VII. Agent Theories Architectures and Languages, 7th International Workshop* (ATAL 2000), C. Castelfranchi, Y. Lesperance (Eds.). LNCS Vol. 1986, Springer Verlag, Berlin, 2001.

DeLoach, S.A., Matson, E.T., and Li, Y., 2003. Exploiting Agent Oriented Software Engineering in the Design of a Cooperative Robotics Search and Rescue System. *The International Journal of Pattern Recognition and Artificial Intelligence*, 17 (5):817-835.

Grosz, B., and Kraus, S., 1996. Collaborative Plans For Complex Group Action. *Artificial Intelligence* 86(2):269-357.

Jennings, N. R. 1993. Commitments and Conventions: The Foundation of Coordination in Multiagent Systems. *The Knowledge Engineering Review*, 8(3):223-250.

Jennings, N.R., 1995.Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions. *Artificial Intelligence*, 75(2):195-240,

Kinny, D., Ljungberg, M., Rao, A. S., Sonenberg, E., Tidhar, G. and Werner, E., 1992. Planned Team Activity. In *Artificial Social Systems - Selected Papers from the Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-92),* Castelfranchi, C. and Werner, E. eds. 226-256. Vol 830 LNAI, Springer-Verlag.

Lawrence, P.R., and Lorsch, J.W., *Organization and Environment: Managing Differentiation and Integration*, Division of Research, Graduate School of Business Tambe, M. "Towards flexible teamwork". *Journal of Artificial Intelligence Research*, 7:83--124, 1997.

Matson, E. and DeLoach, S.A., 2003. An Organization-Based Adaptive Information System for Battlefield Situational Analysis. In *Proceedings of the International* Conference on Integration of Knowledge Intensive Multi-Agent Systems: KIMAS'03: Modeling, Exploration, and Engineering, 46-51. IEEE.

Matson, E. and DeLoach, S.A., 2004. Enabling Intra-Robotic Capabilities Adaptation Using an Organization-Based Multiagent System. In Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA 2004). IEEE.

Turner, R. M., and Turner, E. H., 2001. A Two-Level, Protocol-Based Approach to Controlling Autonomous Oceanographic Sampling Networks. IEEE Journal of Oceanic Engineering, 26(4):654-666.