Separating Application-Specific and Organizational Coordination Issues during Multi-Agent Organizational Design and Instantiation*

Mark Sims, Daniel Corkill, and Victor Lesser
University of Massachusetts
Multi-Agent Systems Laboratory
{msims,corkill,lesser}@cs.umass.edu

UMass Computer Science Technical Report 04-98

November 19, 2004

Abstract

The ability to create effective multi-agent organizations is key to the development of larger, more diverse multi-agent systems. Organizational control provides long-term organizational goals, roles, and responsibilities as guidelines for each agent. Organizational design and instantiation is the process that accepts a set of organizational goals, performance requirements, agents, and resources and assigns responsibilities and roles to each agent. We present a prescriptive organizational design and instantiation process for multi-agent systems. An important aspect of our approach is the separation of application-specific organizational issues from more generic organizational coordination mechanisms. We describe our model of organizational design and our search process and provide examples of how it operates. We also present example organizations generated by our automated system for the distributed sensor network domain under different environmental characteristics and performance requirements.

*Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory Air Force Materiel Command, USAF, under agreement number #F30602-99-2-0525. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. This material is also based upon work supported by the National Science Foundation under Grant Nos. IIS-0004112 and IIS-9988784. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory, NSF, Air Force Office of Scientific Research, or the U.S. Government.

1 Introduction

The ability to create effective multi-agent organizations is key to the development of larger, more diverse multi-agent systems. Organizational control provides long-term organizational goals, roles, and responsibilities as guidelines for each agent. These guidelines reduce the complexity of each agent's operational decision making, lower the cost of distributed resource allocation and agent coordination, help limit inappropriate agent behavior, and reduce communication requirements [2]. To design an organization requires that both application-specific and more generic organizational coordination knowledge be applied to organizational goals, performance requirements, and environmental information in order to generate organizational responsibilities that each agent elaborates into appropriate operational behaviors.

To date, explicitly designed multi-agent organizational structures have been handcrafted, sometimes assisted by automated template expansion [17] or computed adjustments made to a pre-determined structure [16]. This article describes work on developing a fully automated organizational design and instantiation process. With application-specific information, such as organizational goals and agent descriptions, the process first finds groups of agents for each goal with the combined resources to achieve that goal. It then adds more generic organizational structures that enable the agents to coordinate their actions with regard to their joint goals. The process also reserves resources within agents to enable the dynamic formation of teams where statically defined structures are not suitable. In addition to the design process we describe a prototype system that uses the process to create appropriate, yet substantially different, organizational forms when given different requirements and environmental expectations. One important aspect of our approach is the separation of application-specific organizational knowledge from more generic organizational coordination mechanisms. This separation allows the reuse of organizational coordination mechanisms across a wide range of problem domains and environmental situations.

The multi-agent organizational design and instantiation problem is summarized as follows. The input consists of application-specific organizational goals, environmental conditions, performance requirements, possible roles, agents, and resources. The output is an assignment of both application-specific and organizational coordination roles and responsibilities to each agent such that the performance requirements are satisfied and the organization operates effectively over anticipated environmental conditions. To solve this problem autonomously, we have developed the knowledge-based design process illustrated in Figure 1 and described in Section 2.

Before continuing, it is important to distinguish organizational from operational control. Organizational responsibilities represent long-term guidelines while operational control involves short-term agreements among agents to perform specific activities. Our process does not pertain to operational activities. Rather than describe in detail how particular operational decisions are made, the organizational design process ensures that resources and coordination mechanisms exist for agents to make efficient operational decisions during the life of the organization.

Our approach exploits a separation we have observed between application-specific and organizational coordination issues. The former, shown on the left side of Figure 1, involves decomposing high-level organizational goals for the application into simpler

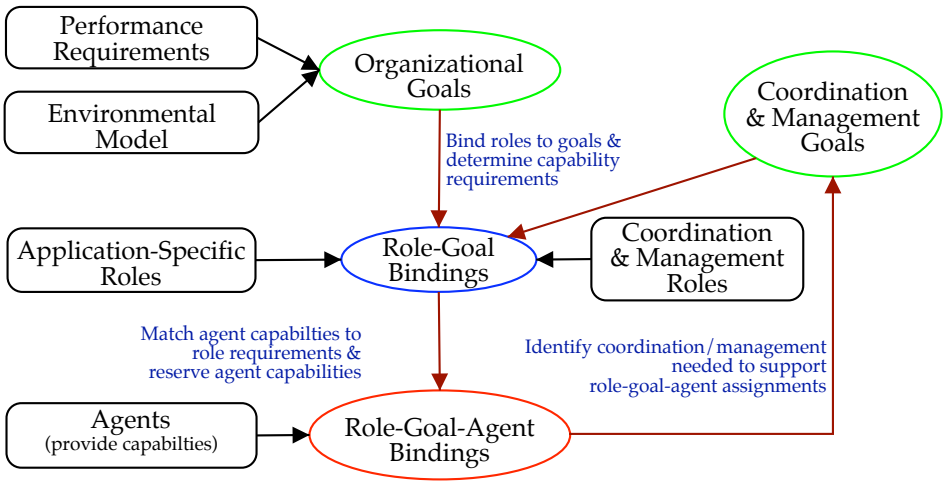


Figure 1: Organizational Design Process

organizational subgoals and then binding them to application-specific roles. The latter, shown on the right, pertains to the coordination mechanisms that are needed when multiple agents are required to perform those roles jointly. The result is a set of bindings for each agent to both application-specific and coordination-specific roles as illustrated in Figure 2. The bindings specify not only the roles the agent is bound to, but also the other agents and their roles that the agent sends information to and receives information from. The figure shows two of agent S24's application-specific roles, RADARSCANNER and FUSER, and one of its coordination roles, SUBORDINATE, which refers to its relationship to an agent performing a MANAGER role.

Consider an agent-based distributed sensor network (DSN). An application-specific organizational goal is to track vehicles with an accuracy of 10 feet and a maximum detection delay of 3 seconds. The environmental model gives the expected traffic volume, spatial density, arrival rate, and vehicle movement. Available roles are radar-based scanning and data processing. While the roles suit varied scenarios, the best coordination mechanism for agents playing them depends on a number of factors. If only a few agents are necessary to cover the monitored area, a peer-to-peer mechanism may be best. If many agents are required, vehicles arrive frequently, and scanning resources are scarce, a multi-level hierarchy may be appropriate.

Our intuition is that organizational coordination knowledge often transcends applications. Thus, a general-purpose, automated organizational design and instantiation system can include generic coordination knowledge. Ideally, the developer would need only supply information about the application to enable the system to determine an organizational structure. This separation allows us to take a prescriptive, knowledge-based approach to organizational design and instantiation that does not require the specification of the coordination mechanisms that will be used in the organization.

Past work in multi-agent organizational design either has been purely descriptive, such as organizational ontologies [6], has used predetermined organizational forms [17],

Agent S24 (82.5, 52.5)
RADARSCANNER→SCAN((62.5,32.5),40,40)
TO: VERIFIER S22
FUSER→FUSE((45,60),45, 30)^T
TO: FOCUSSEDRADAR S22 S24 S23 S18 ...
TO: VERIFIER S22
FROM: FOCUSSEDRADAR S22 S24 S23 ...
FROM: HANDLER S22
SUBORDINATE→COORDGOAL(SCAN)
TO: MANAGER S22
FROM: MANAGER S22
...

Figure 2: Example application-specific and coordination bindings for a single agent resulting after the organizational design process.

or has focused on specifying a specific organizational design, not searching for one [5, 12]. In our work, after the application's features are specified, the system finds organizational structures based on domain-independent coordination knowledge. The work of So and Durfee [16, 15] comes closest to ours. With a model based on the task environment, organizational structure, and performance metrics, they explored how to choose an organizational structure for a given problem. However, they used only hierarchical coordination structures and were primarily concerned with making span-of-control decisions.

Other multi-agent work has dealt with agent coordination but emphasized operational over organizational issues. STEAM [22] provides a hierarchical, role-based framework for the quick formation of agent teams and coordination between them. As such, it is a coordination mechanism that contributes to the automated system's store of knowledge. Similarly, GPGP [14, 4] provides a family of coordination mechanisms, each of which fits within the scope of the automated designer's knowledge.

Several approaches to organizing large groups of agents utilize emergent or bottom-up techniques [24, 23, 19, 21] for self-organization. While there are certainly situations in which such methods are appropriate, time constraints may not allow the self-organization processes to unfold. Also, the quality of an emergent organization may be less than that of a carefully designed one [7, 2].

Finally, Dastani's model for matching agents to roles based on the goals they can achieve [3] has some similarities to ours. However, that work is aimed at enabling agents to enact roles as they enter open agent societies. We are interested in assigning agents to roles so that they may function together as a coherent organization.

The remainder of the article is organized as follows. Section 2 describes our model and the design and search processes. Section 3 provides examples of organizational designs generated by a prototype designer for a DSN under various environmental conditions and performance requirements. We conclude and describe future work in Section 4.

Environmental Model	
maxNewArrivals	10
maxTracks	10
maxVelocity	$20mph$
vehicleWidth	$3'$
(x,y)	(0,0)
length	$90'$
width	$90'$

Performance Requirements	
Detect Delay	$3sec$
Track Resolution	$10'$

Figure 3: Example environmental model

2 Model and Design Process

2.1 Application-Specific Inputs

Referring to the left side of Figure 1, the environmental model M gives the expected environmental features over time and is represented as a set of attribute-values pairs:

$$M = \{\langle f_i, v_{f_i} \rangle\} \quad (1)$$

where f_i is a user specified, domain-specific environmental feature and $v_{f_i} \in \mathbf{R}$. The set of performance requirements Q specifies the requirements that the organization must meet to satisfy the organizational goals. Similar to the environmental model, Q is a set of attribute-value pairs:

$$Q = \{\langle q_i, v_{q_i} \rangle\} \quad (2)$$

where q_i is a feature and $v_{q_i} \in \mathbf{R}$ is its value.

Figure 3 shows an environmental model and performance requirements for the simplified version of the DARPA EW Challenge Problem DSN [10] we refer to throughout the article. In it agents that control radar-based scanners cooperate to track vehicles moving through a rectangular region. The environmental model indicates the expected traffic volume, spatial density, arrival rate, etc. In this example, assume the performance requirements are to track all vehicles with 10 feet of accuracy and a detection delay of at most 3 seconds.

Returning to Figure 1, an organizational goal g is a high-level, long-term objective. We represent organizational goal decomposition as a tree T whose nodes are goals and edges represent subgoal relations. Figure 4 shows a goal tree for our example DSN. The root MONITOR decomposes into subgoals for detecting and tracking vehicles. Similarly, DETECT and TRACK can be further decomposed. Under DETECT, the child SCAN pertains to scanning the monitored area for new vehicles, VERIFY to determining if a detection from the SCAN goal is actually a new vehicle, and HANDLE to setting up activities associated with a new vehicle detection, in this case tracking it. The FUSE goal pertains to fusing data from track updates from the UPDATE goal.

The root goal of the tree is parameterized by the environmental model and performance requirements. Subgoals inherit their parents' parameters unless the developer

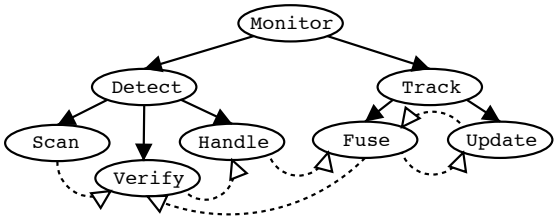


Figure 4: Example DSN goal tree and associated communication graph (dotted edges)

Goal	*TAL*
SCAN	Scanning
VERIFY	Verifying
HANDLE	Handling
FUSE	Fusing
UPDATE	Updating

Figure 5: To-be-assigned lists of the leaf goals in Figure 4.

specifies otherwise. Each goal g also has a to-be-assigned list, TAL, of responsibilities. A leaf goal is *satisfied* if agents bound to it perform the responsibilities in its TAL within the performance requirements on it. A non-leaf goal is satisfied if all of its children are satisfied.

Figure 5 shows the to-be-assigned lists of the leaf goals from Figure 4. Although the goals in our example have single responsibilities in their TALs, in general a goal will entail multiple responsibilities. Figure 6 shows the parameters and TAL of the goal SCAN. SCAN inherits all parameters except $maxTracks$ and $TrackResolution$.

As in traditional planning, where goal decomposition continues until subgoals can be achieved by primitive actions, organizational goal decomposition continues until the TALs of subgoals can be fulfilled by assigning roles. Unlike planning actions, however, roles are atemporal "job descriptions" lasting throughout the organization's lifetime. Each role r_i has an assignable-list AL_i of responsibilities that it can perform and a set of user-defined functions. These functions include a real-valued quality function qf_i indicating how well the role achieves a goal, a set of requirement functions F_i that specify constraints that must be met by agents performing the role when it is bound to a goal, and a distribution function D_i specifying a procedure for how the role when bound to a goal can be distributed among a group of agents. We define the set of available application-specific roles R as

$$R \quad = \quad \{r_i\} = \{\langle AL_i, qf_i, F_i, D_i \rangle\} \; . \tag{3}$$

Figure 7 shows the roles and their assignable lists available in the DSN example. RADARSCANNER is a general purpose scanning role whose primary purpose is to perform sweeps of an area for new vehicle detections. As such its AL contains the TAL *scanning* from the leaf goal SCAN. Also, because RADARSCANNER could be

SCAN((x,y), length, width, maxNewArrivals,
maxVelocity, vehicleWidth, detectDelay)
TAL: Scanning

Figure 6: Parameters and to-be-assigned list of SCAN goal.

used in some applications to track vehicles rather than just detect new ones, its AL contains the TAL *updating* from the goal UPDATE as well. FOCUSSEDRADAR is a directed scanning role specifically for sending vehicle track updates; however, since its information could be used for new vehicle detections, its AL also contains both *updating* and *scanning*. Each remaining role has only a single element in its AL.

For each role, the quality function qf_i and the requirement functions in F_i are dependent on goal parameters (represented by $P_{\{X\}}$ in Figure 7 where $\{X\}$ indicates the first letter of one of the goals in Figure 4) and D_i is a function of the parameters of the goal the role is bound to and the set of available agents A. More specifically, any role whose assignable list contains a goal's TAL may be bound to that goal. However, since this may be true for multiple roles, the quality functions of the roles help to determine which role is the most suitable for a goal given that goal's parameters. For instance, since RADARSCANNER and FOCUSSEDRADAR contain the same elements in their ALs (see Figure 7), by comparing their quality functions, we can determine which is more likely to be useful in a given environment before considering agent bindings. This serves to prune the search space early in the design process by focussing the search on finding agents able to perform the role most likely to satisfy the goal effectively.

While the quality functions are useful in selecting roles that satisfy goals before agents are bound to them, the requirement functions in the set F_i specify requirements that must be met by agents performing those roles when bound to goals. For instance, as shown in Figure 9, when RADARSCANNER is bound to SCAN, the requirement function for RADARSCANNER given the parameters of SCAN, determines how often the region must be scanned to guarantee vehicle detections within the acceptable track delay. This is important because it specifies not only what individual agents must do, but also what the combined behavior of a group of agents must be if no single agent has the capabilities to perform the role individually. In other words, any coordination mechanism used to coordinate the agents performing the role jointly should ensure that the group behavior meets the requirements specified by the requirement functions.

A role's distribution function D_i specifies a procedure for how to divide a group of agents to satisfy a role jointly assuming they could be coordinated properly. A simple distribution function for RADARSCANNER, for example, would specify that the area over which the role is responsible be subdivided among the set of agents such that the agents are responsible for potentially overlapping subareas.

In addition to specifying a goal tree and roles to satisfy the leaves, it is necessary to specify how information is to flow among goals since certain goals require information from others. We represent such relationships as a directed communication graph $G = (L, E)$ where L is the set of leaf goals in T and E is the set of edges between them. An edge (u, v) exists if information must flow from goal u to goal v. The dotted edges between goals in Figure 4 show the communication graph for our DSN. For example,

Role	*AL*	qf_i	F_i	D_i
RADARSCANNER	Scanning, Updating	P_S,P_U	P_S	P_S, A
FOCUSSEDRADAR	Updating, Scanning	P_U,P_S	P_U	P_U, A
VERIFIER	Verifying	P_V	P_V	P_V, A
HANDLER	Handling	P_H	P_H	P_H, A
FUSE	Fusing	P_F	P_F	P_F, A

Figure 7: Application-Specific Roles for the DSN example showing each role's assignable list and the parameters to each of the functions in Equation 3. $P_{\{X\}}$ represents the parameters of a goal where $\{X\}$ indicates the first letter of one of the goals in Figure 4. A is the set of agents.

suppose an edge exists in the communication graph from g_1 to g_2 and that agent sets A_1 and A_2 are bound to each respectively. If the goals are spatial in character, not every agent in A_2 necessarily needs information from every agent in A_1. To represent this, the parameters of each spatially defined goal specify the area the goal is responsible for. Thus, a goal requires information from another only if the information pertains to the goal's area. As seen in Figure 6, the area is specified in the goal's parameters. As we will see below, after the responsibility of handling a spatial goal is distributed among a set of agents, each agent becomes responsible for a subregion of the whole represented by a subgoal with the subregion specified in its parameters. An agent bound to such a subgoal will then send information to another agent bound to a different subgoal if an edge exists between the two subgoals' parents and the subregions specified for each agent overlap.[1] We see an example of responsibility for subgoals in Figure 2 where Agent S24 is responsible for scanning in the subregion with the top-left corner at coordinates (62.5, 32.5) and a length and width of forty feet each while it is responsible for fusing data pertaining to the area with top-left corner at (45,60) and length 45 feet and width 30 feet. As part of its RADARSCANNER role, S24 sends information to Agent S22 which acts as the verifier responsible for the area encompassing S24's scanning area.

Finally, $A = \{a_i\}$ is the set of agents available to the organization. For a_i we specify a set ϕ_i of features such as its location, plus a set $\rho_i = \langle r_k, d_k, m_k \rangle$ of each role r_k that the agent is able to play, the percent drain d_k on the agent's resources caused by r_k, and the number of messages per time m_k the agent sends during its operational performance of r_k (m_k may be a function). We also specify a set $C_i = \{\langle c_j, v_{c_j} \rangle\}$ of capabilities, where c_j is a capability and $v_{c_j} \in \mathbf{R}$ is its value. Thus, $a_i = \langle \phi_i, \rho_i, C_i \rangle$.

[1] In this work, because we have been concerned with a distributed sensor network, we have assumed that goals have spatial characteristics. The parameterization of goals, therefore, explicitly includes their spatial information, and the connections among a goal's subgoals are made based on their spatial character. It is easy to imagine goals for which spatial dimensions do not apply and where the interdependencies among subgoals are based on other features. As we gain experience by applying the design process to more domains, we will abstract the parameterization of goals further to make them more domain-independent.

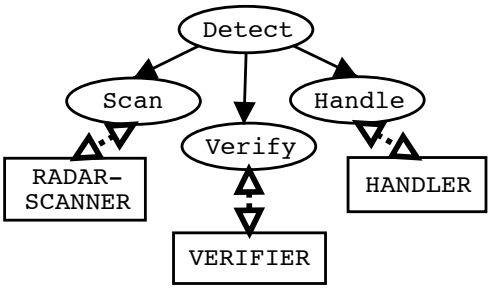


Figure 8: Subtree of the goal tree in Figure 4 with roles bound to each leaf goal.

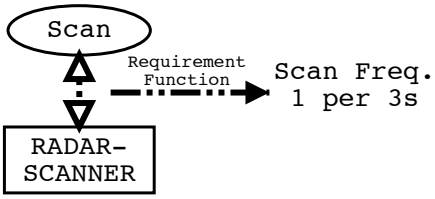


Figure 9: RADARSCANNER's requirement function generates the scan frequency requirement for the RADARSCANNER→SCAN binding.

2.2 Application-Specific Binding

With the above input, the design process attempts to assign application-specific roles to organizational leaf goals to form role-goal bindings. As discussed in Section 2.1, any role whose AL contains a goal's TAL may be bound to that goal while the quality functions of roles allow the process to make decisions about which roles to select. Figure 8 shows a subtree of the organizational goal tree in our DSN example with one role bound to each leaf goal. In this case RADARSCANNER was chosen over FOCUSSEDRADAR to be bound to the goal SCAN.

Binding a role to a goal produces requirements as specified by the role's requirement functions, F_i. As Figure 9 shows, if the RADARSCANNER→SCAN binding is instantiated, RADARSCANNER's application-specific requirement function generates the scan frequency necessary to meet the performance requirement on new vehicle detections. We define the set of role-goal bindings within an organization as:

$$RGB \quad = \quad \{\langle r_i, g_j, \mu_k \rangle\} \tag{4}$$

where $r_i \in R$ and g_j is a leaf goal such that $TAL_j \subseteq AL_i$, and $\mu_k = \{\langle \mu_h, v_{\mu_h} \rangle\}$ is a set of requirement attribute-value pairs determined by r_i's requirement function parameterized by g_j. For RADARSCANNER→SCAN, μ_h and v_{μ_h} specify the scan frequency that must be maintained.

Next the process binds agents to each role-goal binding. Using the agents' capabilities and the roles' distribution functions, the design process identifies agents that meet the requirements of a role-goal binding to form a set of role-goal-agent bindings. The particular binding specifies the role the agent is bound to, the decomposed sub-goal it

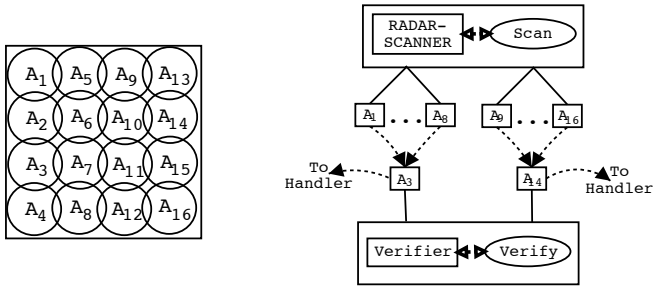


Figure 10: Role-goal-agent bindings for RADARSCANNER→SCAN and VERIFIER→VERIFY. The dotted arrows show how information flows between the bindings.

is responsible for, and the sets of agents it receives information from and sends information to. Continuing with the example, Figure 10 illustrates role-goal-agent bindings for the RADARSCANNER→SCAN and VERIFIER→VERIFY role-goal bindings for a set of sixteen, homogeneous sensor agents. In the distributed sensor network agents have fixed locations and can scan circular areas with a fixed radius (the overlap of the sensors' viewable areas is greater than that depicted). Figure 10 shows that all sixteen agents are bound to RADARSCANNER→SCAN. If this were not the case, gaps would exist in the coverage. The design process was able to make this decision based on RADARSCANNER's distribution function which specifies what the characteristics of a covering set of sensor agents should be.

Figure 10 shows that Agents 3 and 14 are also bound to VERIFIER→VERIFY in addition to RADARSCANNER→SCAN. The reason that a single agent is not bound to the VERIFY role is that no agent among the available set has the processing capabilities both to scan for new vehicle detections and keep track of all existing tracks in order to verify new ones. The reason more than two agents are not bound to the VERIFY role is that to do so would put extra load on the agents and require more communication since the agents acting as VERIFIERs would need to share information about existing tracks in order to verify new detections accurately. As we will see in Sections 2.5 and 3, the decision to split a role in a particular way can not be entirely specified by a role's distribution function and often relies on heuristics. In this case, Verifier's user-defined distribution function specifies that the role should be divided among agents such that their capacities are not exceeded and that agents performing that role are spatially near the agents they are performing the role for. The decision to limit the number of VERIFIERs to two is an attempt to make a tradeoff between agent loading and inter-agent communication.

The dotted arrows in Figure 10 show the application-specific flow of information between the sets of bindings as determined by the communication graph in Figure 4. Each scanning agent must send information to a verifying agent and since scanning is a spatial activity, different verifiers are responsible for the different groups of scanners. Also, each VERIFIER agent must send information to an agent bound to the goal of handling new detections as specified by the edge between VERIFY and HANDLE.

Note that the dotted arrows do not specify the flow of information pertaining to the organizational coordination of the agents. While the application-specific bindings do provide some level of coordination among agents, organizational coordination structures are still necessary. For instance, the application-specific bindings do not give agents acting as scanners the ability to schedule their scans to provided sufficient coverage. Such coordination bindings are made later in the design process.

We define the set of role-goal-agent bindings of agent $a_i \in A$ as

$$RGAB_{a_i} \quad = \quad \{\langle r_k, g_j, g'_j, f_{g'_j}, t_{g'_j}, T\rangle\} \tag{5}$$

where $r_k \in R$, g_j is a leaf goal, g'_j is a subgoal of g_j as determined by r_k's decomposition method D_k, $f_{g'_j}$ is the set of agents a_i receives information from, $t_{g'_j}$ is the set of agents a_i sends information to, and T is a flag indicating if this binding is a teaming assignment (described below). As discussed in Section 1, Figure 2 shows an example of one agent's role-goal-agent bindings.

2.3 Coordination-Domain Binding

So far, the organizational design process has involved only application-specific information shown in the left half of Figure 1. An advantageous feature of our approach is that the rest of the process can use more domain-independent organizational coordination knowledge to add coordination structures. In general, a role will require multiple agents to fulfill the performance requirements of an organizational subgoal. Not only must role-goal-agent bindings be found, but those agents must be coordinated in performing their roles. The agents bound to RADARSCANNER→SCAN have the necessary capabilities to satisfy the requirements, but unless their scanning is synchronized correctly, holes may exist in the coverage since in the DSN, sensor agents have limited range and at least three are needed to triangulate the position of any vehicle.

Consider Figure 11 which illustrates coordination goal generation and the assignment of coordination roles for the application-specific bindings in Figure 10. Because the RADARSCANNER role is split among a group of agents, as shown in Figure 11a, the agents must be coordinated in their fulfillment of that role. This automatically causes the system to generate a new *coordination goal* that was not part of the original goal decomposition. This new goal must be fulfilled by more domain-independent coordination roles, as shown on right of Figure 1. Possible coordination roles for the sensing agents include: peer-to-peer negotiation of scan schedules or a simple, one-level hierarchy where a manager agent develops the scan schedule for the group. In Figure 11b, a one-level hierarchy is used. In deciding which agents should act as managers, the design process considers the other roles each agent plays and the relative utility of assigning managerial responsibilities to them. In Figure 11b, Agents 3 and 14, which are also VERIFIER agents are chosen to be the managers. This is to minimize the amount of communication by multiplexing verifying and managing within the same agent. As will be seen in Section 3, if balancing agent load is more important than communication usage, it may be better to bind other agents to the managerial role.

A coordination role-goal-binding can, itself, require a set of agents to satisfy it, causing the creation of another higher-level coordination goal, as seen in Figure 11b

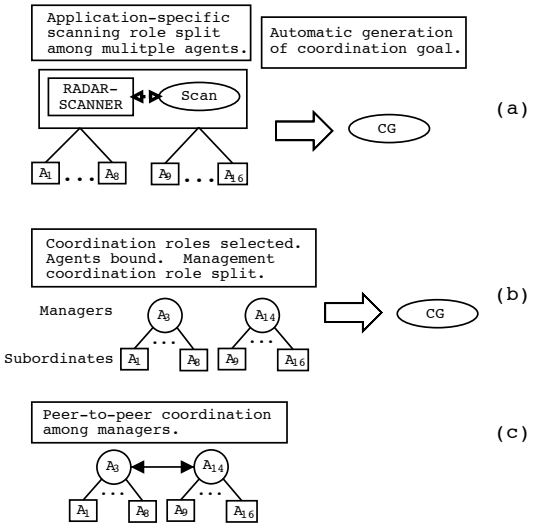


Figure 11: Illustration of coordination goal generation and the assignment of coordination roles. (a) The RADARSCANNER role is split among a group of agents which automatically causes the system to generate a new coordination goal. (b) In this example the system chooses a one-level hierarchy to satisfy the new goal, but since the management role must also be split among a group of agents, it generates yet another coordination goal which (c) in this example is satisfied by a peer-to-peer structure.

where the manager role is divided. In Figure 11c a peer-to-peer mechanism is chosen to satisfy the goal because only two managers are present. In other situations, another level of hierarchy may also have been chosen resulting in a multi-level hierarchy of sensing, middle-manager, and overall manager roles as seen in Figure 14.

2.4 Teams

The role-goal-agent bindings and their parameters specify the long-term structure of the designed organization. Although such bindings are appropriate for long-term organizational goals, more transient goals are better satisfied by teams [6, 22, 18, 20, 13, 1]. Teams, coalitions, and congregations are temporary structures that form to satisfy particular tasks that enter the environment and disband when the tasks are completed. In the DSN, tracking a vehicle might be done by a team whose membership changes as the vehicle moves. Teams are not strictly part of the organizational structure; the assignment of agents to roles associated with the team will be shorter lived than the assignment of agents to roles to satisfy organizational goals. However, teams are not purely operational either, as sufficient resources must be set aside organizationally to allow for generating and participating in teams. Furthermore, when an agent within an organization is participating in a team, its team activities will impact its performance in its other roles. Therefore, the organizational structure must be prepared for team activity by its members.

Our design process does not generate teams. Rather, it ensures that appropriate organizational structures and resources are reserved to form teams as needed. For the DSN, this means finding role-goal-agent bindings for the leaf goals of TRACK and setting the team flag T to true to indicate that agents participate in the role only as needed. A team role resembles an organizational role in that the agent with a team-role will have an expected number and frequency of messages to send and amount of work to do. The agents bound to these roles, however, will only be expected to perform those activities if and when they are called upon to join a team. We must also specify appropriate coordination roles in order to enable teams to form. In this work, we define a TEAMINITIATOR role that is responsible for generating teams operationally. Note that this is not an application-specific role input to the organizational design process. Rather, it is a coordination role that is part of the store of knowledge contained within the organizational design process.

As mentioned in Section 2.2, Figure 2 shows a set of bindings for an agent in the DSN. Each binding specifies which organizational subgoal the agent is bound to and the agents to which it sends information and those from which it receives information. If the role is a teaming assignment such as FUSER→FUSE, it is signified with a superscript T.

2.5 Search and Suitability

In general, multiple roles can satisfy the same organizational subgoal, many agents can be bound to a role-goal binding, and each agent can play multiple roles, making it computationally infeasible to generate every binding. Our prototype system uses organization-design knowledge and heuristics to generate a reasonable set of bindings. For the application-specific portions of the design process, the heuristics use information provided by the developer in the quality, requirements, and decomposition specifications of the roles plus the capabilities of the agents. For example, to evaluate and compare sets of role-goal bindings the system finds the average value of the roles' quality functions given the goals they are bound to. For instance, consider Figure 8 once more. It shows a subtree of the organizational goal tree with roles bound to each leaf goal. In an alternate set of role-goal bindings, FOCUSSEDRADAR, not RADARSCANNER, could be bound to SCAN because FOCUSSEDRADAR also contains *scanning* in its assignable list (see Figure 7). The original set of bindings, however, will have a higher average quality than the other, since RADARSCANNER's quality function will have higher value than FOCUSSEDRADAR'S when bound to scan. This causes the search process to explore organizational candidates with the role-goal bindings shown in Figure 8 before considering candidates that use the other.

To evaluate the communication load on the system, the search process determines the ratio of the average bandwidth required by the agents to perform their roles to the available bandwidth. Again, consider the DSN. As we saw in Section 2.3, it is possible to assign the Manager role to an agent acting as both VERIFIER and RADARSCANNER or to an agent acting solely as a RADARSCANNER. If the Manager role is multiplexed within the same agent as the VERIFIER role and the Manager and VERIFIER are responsible for the same agents, the search process assumes that the agent is able to combine verifying and managing messages to reduce the total bandwidth it requires.

If the Manager role were assigned to an agent acting solely as a RADARSCANNER, the search process assumes that the agent would have to send management messages in addition to its scanning messages and the verifying messages of the VERIFIER. Thus, the former would have a smaller combined communication load than the latter.

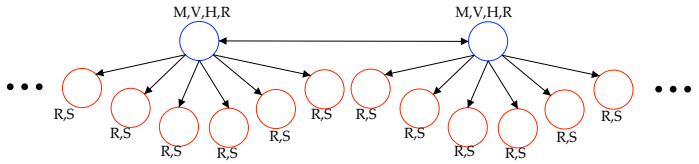
In general the heuristics consider which roles should be bound to the organizational goals, which agents can be bound to particular role-goal bindings, and the computational and communication loading on agents that would result under different assignments. In addition, the search may require some amount of backtracking since initial binding choices may lead to states in which no agent given its current set of roles and capabilities can satisfy the remaining responsibilities.

For coordination goals, the design system goes through a process of finding role-goal-agent bindings similar to the process of finding bindings for organizational goals. The main difference is that the roles available for satisfying the coordination goals and the search heuristics exist within the system as domain-independent knowledge. The interface between the application-specific and organizational coordination-specific roles and goals is a set of parameters. In the current prototype the set is not yet complete. The primary parameter used is the number of agents to be coordinated. For instance, the heuristic for choosing between peer-to-peer and hierarchical coordination mechanisms for a group of n agents assumes that the number of messages necessary for the former is $O(n^2)$ and $O(n)$ for the latter. Thus, only for very small numbers of agents will the system choose a peer-to-peer mechanism. Related to the number of messages is the assumption that it is better for agents who must communicate to be spatially near one another.

Another heuristic used in making coordination decisions has to do with the amount of time available to perform a task. If the time is small, the system is more likely to add a level of hierarchy to an existing hierarchy. The assumption is that high-level managers can notice and correct for inefficiencies not noticed by lower-level managers and may be worth the overhead of a more complicated structure. This will be seen in Section 3 for the DSN when the acceptable delay on new vehicle detections is small.[2] In future research, we plan to develop more principled abstractions of application parameters to coordination parameters and heuristics that rely on a detailed evaluation function and a better understanding of the subgoal interdependence. For now we note that mechanisms similar to those in our prototype for structuring organizations are common in the organization design literature [8].

Although the heuristics above should lead to an organization that meets the performance requirements, they do not give enough information to rank candidate organizations that all satisfy the requirements. We must consider other factors to evaluate them. For that it is important to have an organizational evaluation function that is based on user-specified criteria to determine a particular candidate's utility. In on-going work,

[2]Similar to the note in Section 2.1, the heuristics described here make assumptions about the type of interdependencies that exists among organizational subgoals. In particular, because we have focused on a distributed sensor network, we have assumed spatial and temporal interdependencies. In other domains these assumptions may not hold. As we continue our work and apply it to new domains, it is important for us to characterize the interdependencies among subgoals in order to create more generalized parameterizations of application-specific goals. Incorporating a better understanding of subgoal interdependence into the system will enable the system to apply coordination where needed and choose the mechanism appropriate for a given type of interdependency.



Agents	Area	Delay	Com. Cost	Load Cost
36	$90' \times 90'$	$3s$	0.6	0.4
Single-level hierarchy: 6 Managers. Verifier and Handler roles multiplexed within same agent as Manager. Managers coordinate peer-to-peer.				

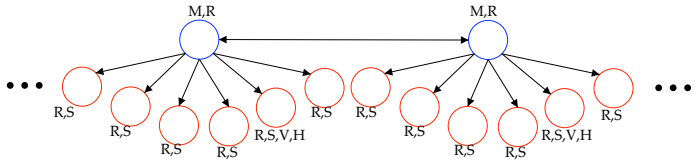
Figure 12: Example organization with the cost of communication greater than that of agent load. The labels in the figures refer to the roles present in the organization. M stands for manager, S subordinate, V verifier, H handler, and R radar scanner.

we are developing a detailed evaluation capability to evaluate fully specified organizations and to prune the search through partially complete ones. For now, we rely on simple utility criteria stemming from the relative costs of agent load and communication. Using user-defined weights, the utility function is a weighted sum of the ratio of the required bandwidth to the available bandwidth and the average fraction of the resource usage of each agent.

3 Example Organization Designs

We present four example organizational designs generated by our automated system on the goal tree and communication graph in Figure 4, the parameters in Figure 3, and the roles in Figure 7. We varied the input along several dimensions: size of the area monitored, number of agents, value of the acceptable detect delay, and the relative costs of communication and agent load. In all cases the agents we used were evenly spaced throughout the region, each with identical features, roles they can be bound to, and capabilities. Figures 12-15 summarize the results. Each organization in the figures took only a few seconds to generate because the heuristics of the search process substantially prune the search space. For instance, as described in Section 2.1, the search process uses the roles' quality functions so as not to expand partial organizational candidates that use low-quality role-goal bindings. Similarly, the process takes a greedy approach to binding agents to role-goal binding. This can result in the need for backtracking, but in the examples presented here it did not. In continuing work, we are investigating how to explore the space more fully by evaluating a greater number of permutations of agents within roles and roles bound to goals.

The first scenario shown in Figure 12 involved 36 agents in a $90' \times 90'$ rectangular area, an acceptable detect delay of 3 seconds, and the cost of communication greater than that of agent loading. The result was a single-level hierarchy with 6 managers each managing 6 agents. The managers coordinated among themselves using a peer-to-peer mechanism. In order to minimize communication, there were 6 verifying and handling



Agents	Area	Delay	Com. Cost	Load Cost
36	$90' \times 90'$	$3s$	0.4	0.6
Single-level hierarchy: 6 Managers. Verifier and Handler roles not multiplexed with Manager. Managers coordinate peer-to-peer.				

Figure 13: Example organization with the cost of communication less than that of agent load. The labels in the figures refer to the roles present in the organization. M stands for manager, S subordinate, V verifier, H handler, and R radar scanner.

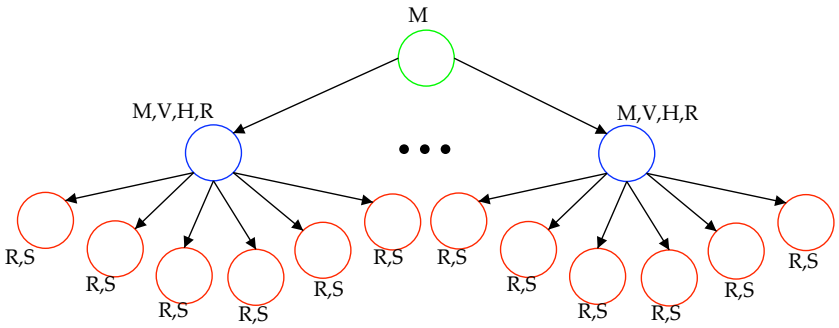
roles each multiplexed within the same agents as the managing roles. This organization corresponds closely to the hand-crafted structure used for the EW Challenge Problem [10] where communication cost was a major concern. The performance of this organizational form relative to others was recently tested experimentally [9, 11]. Also, in this scenario and the others, the FUSER and FOCUSSEDRADAR roles were set as team roles with the TEAMINITIATOR role distributed among the HANDLER agents.

Switching the relative costs of communication and load still resulted in a single-level hierarchy as shown in Figure 13, but the verifying and handling roles were no longer multiplexed within managers. They were distributed to separate agents to minimize load. In effect because communication was inexpensive, the organization could afford to use more communication in order to balance the computational load among the agents.

For the third scenario, we used the same costs as in the first, but reduced the acceptable track delay to 2 seconds. This time the generated organization was a two-level hierarchy with 6 mid-level managers and 1 upper-level manager to coordinate them as show in Figure 14. At first this may seem counter-intuitive since increasing the level of hierarchy can often introduce delays. However, in this problem with a small acceptable delay on new detections, it is critical that the scanning agents have tightly synchronized scan-schedules. Because producing a shared scan-schedule can be done in advance of detection activities, the design system added a second level of hierarchy in order to resolve scan-schedule conflicts among the managers in a centralized fashion.

In the last scenario, the parameters were also the same as in the first run except that we increased the number of agents to 100 and the size of the region to $150' \times 150'$. In this case the system generated another two-level hierarchy as seen in Figure 15 this time with nine managers and two upper-level managers which coordinate using a peer-to-peer mechanism. The extra-level was added since to coordinate the nine managers in a peer-to-peer fashion would have incurred greater communication overhead.

We were pleased that our design system produced such appropriately different organizational forms given only changes to the environmental characteristics and performance requirements. These results confirm for us the usefulness of our approach in



Agents	Area	Delay	Com. Cost	Load Cost
36	$90' \times 90'$	$2s$	0.6	0.4
Two-level hierarchy: 6 mid-level Managers. Verifier and handler roles multiplexed within mid-level Managers. One upper-level Manager to coordinate mid-level Managers.				

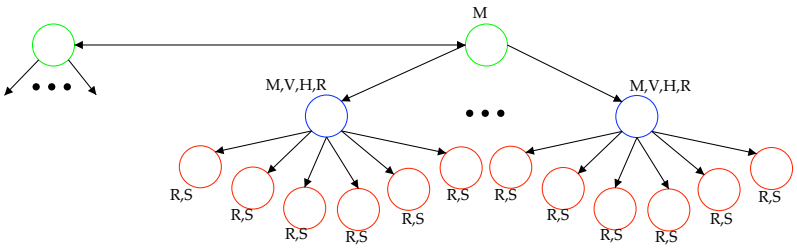
Figure 14: Example organization with reduced acceptable track delay. The labels in the figures refer to the roles present in the organization. M stands for manager, S subordinate, V verifier, H handler, and R radar scanner.

generating organizational forms without pre-specified organizational information.

4 Conclusions and Future Work

We believe that the prescriptive, knowledge-based organizational design process we have presented has great promise. It relies on a separation between application and organizational coordination issues to generalize coordination mechanisms across domains, requiring a developer only to supply problem-specific organizational information. The results from our prototype system show that through this process we are able to design organizations of different forms by varying performance requirements and environmental characteristics. We believe this is the first work to do so.

We have identified several areas of future work stemming from the initial research presented here. First, we will develop further the internal evaluation capability of our system beyond the current simple weighted sum of agent load and communication utility criteria. The new evaluation mechanism must rank candidate organizations given the set of agent bindings, performance requirements, and more detailed evaluation criteria specified by the developer. We also hope to apply the evaluation capability to partial bindings in order to prune more quickly the search for a suitable organization. Another long-term goal is that in addition to evaluating generated organizations, we would like the system to suggest what additional resources and capabilities, if they were provided, would have supported a better organization. In addition to the internal evaluation capability, it is important for us to have an external means of evaluating the organizational designs produced by our system. Such a mechanism must include detailed analysis of an organization's performance and simulation results and will be



Agents	Area	Delay	Com. Cost	Load Cost
100	$150' \times 150'$	$3s$	0.6	0.4
Two-level hierarchy: 9 mid-level Managers. Verifier and handler roles multiplexed within mid-level Managers. Two upper-level Managers to coordinate mid-level Managers. Upper-level Managers coordinate peer-to-peer.				

Figure 15: Example organizations.

especially important as we use the design process in new application domains with which we have less experience.

We must also improve the search and backtracking process to explore the space of organizations more effectively and clarify the knowledge engineering process for domains to simplify the developer's job of specifying domain-specific organizational information. Finally, we must continue to refine our understanding of generic coordination knowledge so as to parameterize the coordination roles more appropriately. Part of this will involve understanding the distinguishing features of organizational goals and how those features relate to the mechanisms available to coordinate the agents bound to those goals. In part this will involve a greater understanding of aspects such as how resource contention, the number of agents bound to a goal, and the interdependency among agents and subgoals interrelate. Finally, although our work currently includes a model of task duration and resource requirements, we do not consider task, communication, or agent failure as part of the expected organizational environment. In future work, we plan to extend our design system to incorporate these factors.

References

[1] Christopher H. Brooks and Edmund H. Durfee. Congregation formation in multiagent systems. *Journal of Autonomous Agents and Multiagent Systems*, 7:145–170, 2003.

[2] Daniel David Corkill. *A Framework for Organizational Self-Design in Distributed Problem-Solving Networks*. PhD thesis, University of Massachusetts, Amherst, Massachusetts 01003, February 1983.

[3] Mehdi Dastani, Virginia Dignum, and Frank Dignum. Role-assignment in open agent societies. In *Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 489–496. ACM Press, 2003.

[4] K. Decker and V. Lesser. Generalizing the partial global planning algorithm. *International Journal on Intelligent Cooperative Information Systems*, 1(2):319–346, June 1992.

[5] Jacques Ferber, Oliver Gutknecht, and Fabian Michel. From agents to organizations: An organizational view of multiagent systems. In *Proceedings of the Fourth International Workshop on Agent Oriented Software Engineering (AOSE03)*, pages 214–230. Springer Verlag, 2003.

[6] Mark S. Fox, Mihai Barbuceanu, Michael Gruninger, and Jinxin Lin. An organization ontology for enterprise modelling. In Michael Prietula, Kathleen Carley, and Les Gasser, editors, *Simulating Organizations: Computational Models of Institutions and Groups*, pages 131–152. AAAI/MIT Press, 1998.

[7] Jay Galbraith. *Designing Complex Organizations*. Addison-Wesley, 1973.

[8] Jay R. Galbraith. *Organization Design*. Addison-Wesley, 1977.

[9] Bryan Horling, Roger Mailler, and Victor Lesser. A Case Study of Organizational Effects in a Distributed Sensor Network. In *Proceedings of the International Conference on Intelligent Agent Technology (IAT 2004)*, Beijing, China, September 2004.

[10] Bryan Horling, Roger Mailler, Jiaying Shen, Regis Vincent, and Victor Lesser. Using Autonomy, Organizational Design and Negotiation in a Distributed Sensor Network. In Victor Lesser, Charles Ortiz, and Milind Tambe, editors, *Distributed Sensor Networks: A multiagent perspective*, pages 139–183. Kluwer Academic Publishers, 2003.

[11] Bryan Horling, Roger Mailler, Mark Sims, and Victor Lesser. Using and Maintaining Organization in a Large-Scale Distributed Sensor Network. *Proceedings of the Workshop on Autonomy, Delegation, and Control (AAMAS03)*, July 2003.

[12] Jomi Fred Hubner, Jaime Simao Sichman, and Olivier Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. In *Proceedings of the 16th Brazilian Symposium on Artificial Intelligence*, pages 118–128. Springer-Verlag, 2002.

[13] M. Klusch and A. Gerber. Dynamic coalition formation among rational agents. *IEEE Intelligent Systems*, 17(3):42–47, May/June 2002.

[14] V. Lesser, K. Decker, T. Wagner, N. Carver, A. Garvey, B. Horling, D. Neiman, R. Podorozhny, M. NagendraPrasad, A. Raja, R. Vincent, P. Xuan, and X.Q. Zhang. Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143, July 2004.

[15] Young pa So and Edmund H. Durfee. Designing tree-structured organizations for computational agents. *Computational and Mathematical Organization Theory*, 2(3):219–246, 1996.

[16] Young pa So and Edmund H. Durfee. Designing organizations for computational agents. In *Simulating Organizations: Computational Models of Institutions and Groups*, pages 47–64. AAAI Press/MIT Press, 1998.

[17] H. Edward Pattison, Daniel D. Corkill, and Victor R. Lesser. Instantiating descriptions of organizational structures. In Michael N. Huhns, editor, *Distributed Artificial Intelligence*, Research Notes in Artificial Intelligence, chapter 3, pages 59–96. Pitman, 1987.

[18] Tuomas Sandholm and Victor Lesser. Coalitions among computationally bounded agents. *Artificial Intelligence, Special Issue on Economic Principles of Multi-Agent Systems*, 94(1):99–137, January 1997.

[19] David Servat and Alexis Drogoul. Combining amorphous computing and reactive agent-based systems: a paradigm for pervasive intelligence? In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 441–448. ACM Press, 2002.

[20] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1–2):165–200, 1998.

[21] Mark Sims, Claudia Goldman, and Victor Lesser. Self-organization through bottom-up coalition formation. In *Proceedings of Second International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2003)*, 2003. To appear.

[22] 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.

[23] H. Van Dyke Parunak. Making swarming happen. In *Proceedings of the Conference on Swarming and Network Enabled Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance (C4ISR)*, January 2003.

[24] H. Van Dyke Parunak and Sven Brueckner. Entropy and self-organization in multi-agent systems. In *Proceedings of the fifth international conference on Autonomous agents*, pages 124–130. ACM Press, 2001.