Separating Domain and Coordination in 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

Abstract

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. We also present example organizations generated by our automated system for the distributed sensor network domain under different environmental characteristics and performance requirements.

1. Introduction

The ability to create and maintain effective multi-agent organizations is key to the development of larger, more diverse multi-agent systems. Through organizational control, long-term organizational goals, roles, and responsibilities are developed and maintained to serve as guidelines for making detailed operational control decisions by individual agents. These organizational 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]. Designed organizations are created by applying both generic and application-specific organization-design knowledge, organizational goals and performance requirements, and task-environment information to generate explicit organizational responsibilities that are then elaborated by the individual agents into appropriate operational behaviors.

To date, multi-agent organizational structures used for control have been hand-crafted, sometimes assisted by automated template expansion [12] or computed adjustments made to a pre-determined structure [11]. In this paper, we describe work on developing an automated organizational design and instantiation system that is able to create appropriate, yet substantially different, organizational forms based on different requirements and task-environment expectations. One important aspect of our approach is **the separation of application-specific organizational knowledge from more generic organizational coordination mechanisms**. This separation will allow the reuse of organizational coordination mechanisms across a wide range of problem domains and environmental situations.

The multi-agent organizational design and instantiation problem can be summarized as follows. Given a problem-domain description of the organizational goals, environmental conditions, performance requirements, possible roles, agents, and resources, assign both problem-domain and coordination roles and responsibilities to each agent such that the organizational performance requirements are satisfied and the organization operates effectively over anticipated environmental conditions. This assignment constitutes the organizational structure. To solve this problem in an automated fashion, we have developed the prescriptive, knowledge-based design process illustrated in Figure 1, which we describe in detail in Section 2.

Before continuing, it is important to clarify the distinction between organizational and operational control and the focus of our work. Organizational roles and responsibilities represent general, long-term guidelines while operational control involves specific short-term agreements among agents to perform specific activities for specific time periods. Our process does not pertain to operational activities. Rather than describe how particular control decisions are made, it ensures that sufficient resources and coordination mechanisms exist to enable agents to make efficient operational decisions throughout the life of the organization.

As mentioned above, our approach makes use of a separation we have observed between the problem-domain and organizational coordination. The former, shown on the left

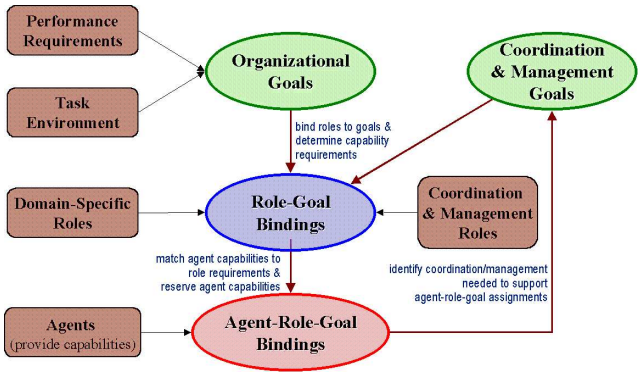

Figure 1. Organizational Design Process

side of Figure 1, involves decomposing high-level organizational goals and matching them to problem-domain roles. The latter, shown on the right, pertains to the coordination mechanisms used by the agents in jointly performing those roles. The result is a set of bindings for each agent to both problem-domain specific and coordination-specific roles as illustrated in Figure 4(c).

As an example, consider a simple distributed sensor network (DSN) application. A problem-domain organizational goal might be to track all vehicles moving within a monitored area with a positional accuracy within 10 feet and a detection delay of at most 3 seconds. The environmental model indicates the expected traffic volume, spatial density, arrival rate, and movement characteristics. The available roles might be radar-based scanning and data processing which are clearly appropriate for a variety of scenarios. The best way to coordinate the agents playing the roles, however, is dependent on a number of factors. If the area to be scanned is small enough that only a few agents are necessary, a peer-to-peer mechanism may be the right choice. If many agents are required, new vehicles arrive frequently, and the scanning resources are scarce, a multi-level hierarchical structure may be more appropriate.

It is our intuition that organizational coordination knowledge transcends the problem domain. Therefore, an automated system can include generic coordination knowledge, requiring the developer to supply information about the problem domain only. The system itself can then use both sets of knowledge in determining an appropriate organizational structure for the agents. Separating problem domain knowledge from coordination knowledge contributes to the field of organizational design in that it allows us to take a prescriptive, knowledge based approach to organizational design and instantiation that does not pre-specify coordination mechanisms.

Past work in multi-agent organizational design has been purely descriptive, such as the organizational ontology of Fox, et. al. [4], or has used predetermined organizational forms as in Pattison, et. al. [12]. In our work only the problem domain features need to be specified; organizational structures are found based on domain-independent coordination knowledge. So and Durfee's work [11, 10] is the closest to ours in that they have a model based on the task environment, organizational structure, and performance metrics and explore the question of how to choose the best organizational structure for a given problem. However, they assume a hierarchical structure and are primarily concerned with making span of control decisions within it.

Still other multi-agent work deals with coordinating agent activity but emphasizes operational issues rather than organizational ones. STEAM [15], for instance, provides a hierarchical role-based framework for the quick formation of agent teams and coordination between them. Within our context, STEAM is an example of the type of coordination mechanism that could exist within the automated system's store of knowledge. Similarly, GPGP [9, 3] provides a family of coordination mechanisms each of which fits within the scope of the automated designer's knowledge.

The remainder of the paper 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.

2 Model and Design Process

2.1 Problem-Domain Inputs

Refer once more to the left side of Figure 1. The environmental model M gives the general expectations of the environment over a period of 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 specify the requirements that must be met by the organization in order for it to satisfy the organizational goals. We represent Q as a set of attribute-value pairs similar to the environmental model:

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

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

Figure 2 shows an example environmental model and set of performance requirements for the DSN example that we will refer to throughout the paper. The example is a simplified version of the EW Challenge Problem domain [7] in

which agents that control radar-based scanners must cooperate to track vehicles moving through a rectangular region. The environmental model indicates the expected traffic volume, spatial density, arrival rate, etc. The performance requirements are to track all vehicles with 10 feet of accuracy and a detection delay of at most 3 seconds.

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 2. Example environmental model

Returning to Figure 1, an organizational goal g is a high-level, long-term objective of an organization. We represent the decomposition of organizational goals in a tree T with root r. The nodes of T are goals and the edges represent subgoal relations. Figure 3 illustrates a goal tree for our example DSN. It shows that the high-level root goal MONITOR can be decomposed into a subgoal for detecting new vehicles and one for tracking detected vehicles. Similarly, DETECT and TRACK can be further decomposed.

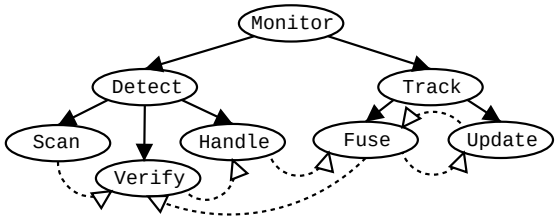

Figure 3. Example DSN goal tree and associated communication graph (dotted edges)

The root is parameterized by the environmental model and the performance requirements. Each other goal inherits the parameters of its parent unless otherwise specified by the developer.

In addition to its parameters and features, each goal g has a to-be-assigned list TAL of responsibilities that need to be assigned to an agent or agents in order for g to be satisfied. We define a goal $g \in L$ where L is the set of leaf goals to be *satisfied* if agents bound to it perform each of the responsibilities in its to-be-assigned list within the performance requirements on it. For all $g \notin L$, g is satisfied if all of its children are satisfied.

Figure 4(a) shows an example of the parameters and to-be-assigned list of the goal SCAN from Figure 3. It inherits all parameters except for maxTracks and Track Resolution. Although SCAN and the other goals in our example have single responsibilities in their to-be-assigned lists, in general a goal will have multiple responsibilities to be fulfilled.

As in a traditional planning system, where goal decomposition continues until the subgoals can be achieved by primitive actions, organizational goal decomposition continues until the to-be-assigned lists of subgoals can be fulfilled by the assignment of roles. Unlike actions in a planning system, however, roles are atemporal responsibilities to be performed throughout the organization's lifetime. Roles are "atomic" job descriptions used to satisfy organizational goals. Each role r_i has an assignable-list AL_i of responsibilities that it can perform, a quality function f_i indicating how well it achieves a goal, a set of requirement functions, F_i, dependent on the parameters of the goals the role may be bound to, and a function D_i specifying how the role when bound to a goal can be distributed among a group of agents. Thus, we define the set of available problem-domain roles R as

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

Figure 4(b) shows the roles and their assignable lists available in the DSN example. As with the goals, although each has only a single responsibility in its AL, in general roles can have multiple responsibilities. For each role, the functions f_i and F_i are dependent on goal parameters (represented by P_X where X indicates one of the goals in Figure 3) and D_i is a function of the parameters of the goal the role is bound to and the set of available agents A (discussed below). F_i for RADARSCANNER, for instance, given the parameters of SCAN determines how often the region must be scanned in order to guarantee that vehicles are detected within the acceptable track delay requirement.

Certain goals require information from other goals. We represent such communication relationships as a directed communication graph $G = (L, E)$ where L is the set of leaf goals in T and E is a set of edges between the leaves such that there exists an edge (u, v) if information must flow from goal u to goal v. The dotted edges between goals in Figure 3 represent the communication graph for our DSN example. Suppose there is an edge 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, it is not necessarily the case that every agent in A_2 needs all of the information from every agent in A_1. We represent this notion within the parameterization of each goal by specifying for each spatially defined goal 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 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 and sends

information to the relevant agents bound to connected goals.

To complete the problem-domain input, let $A = \{a_i\}$ be 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). In addition we 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, $a_i = \langle \phi_i, \rho_i, C_i \rangle$.

2.2 Problem-Domain Matching

With the above input, the design process first attempts to match problem-domain roles to organizational goals to form role-goal bindings, assignments of specific roles to each leaf goal in T. Any role whose assignable-list contains a goal's to-be-assigned list may be bound to that goal. In the DSN example the RADARSCANNER role can be bound to the SCAN goal forming the RADARSCANNER→SCAN role-goal binding. When a role is bound to a goal, it produces requirements as specified by the role's requirement functions. We define the set of role-goal bindings within an organization as a set of triples:

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

where $r_i \in R$ and g_j is a leaf goal of T 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 and its parameters. For RADARSCANNER→SCAN, μ_h and μ_v specify the scan frequency that must be maintained.

The next step in the process is to bind agents to each role-goal binding. Continuing with the RADARSCANNER→SCAN example, the design process identifies agents that can meet the requirements of the role-goal binding according to the agents' capabilities and RADARSCANNER's distribution function 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 is responsible for, and the sets of agents it receives information from and sends information to. Thus, we define the set of role-goal-agent bindings of agent $a_i \in A$ as

$$RGAB_{a_i} \quad = \quad \{\langle r_k, g_j, g'_j, f_{g'_j}, t_{g'_j}, team \rangle\} \tag{5}$$

where $r_k \in R$, g_j is a leaf goal of T, g'_j is a subgoal of g_j as determined by r_k's decomposition method, $f_{g'_j}$ is the set of agents a_i receives information from pertaining to this binding, $t_{g'_j}$ is the set of agents a_i sends information to, and $team$ is a boolean flag indicating that this binding is a teaming role assignment (described below).

2.3 Coordination-Domain Matching

Up to this point, the design process involves problem-domain specific knowledge of goals, roles, performance requirements, agent capabilities, etc., and is shown in the left half of Figure 1. What is advantageous to our approach is that the remainder of the organizational-design process can be addressed using more domain-independent organizational coordination knowledge. In general, a role will require multiple agents to fulfill the performance requirements of an organizational subgoal. In our example, sensor agents have limited range and synchronized scanning by at least three agents is required throughout the coverage area. Not only must role-goal-agent bindings be found as above, but those agents must also 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.

The need to coordinate these agents causes the system to generate a new *coordination goal* that was not part of the original goal decomposition. This organizational-coordination goal must be fulfilled by more problem-domain-independent coordination roles, as shown on the right side of Figure 2. Possible roles for coordinating our set of sensing agents include: peer-to-peer negotiation of scan schedules and a simple, one-level hierarchy where a manager agent (potentially, but not necessarily, one of the sensing agents) develops the scan schedule for the group. A coordination role-goal-binding can, itself, require a set of agents to satisfy it, causing the creation of another higher-level coordination goal. For example, if the span of control of potential manager agents requires the use of multiple managers, the activities of these managers would also need to be coordinated again, potentially using a peer-to-peer or hierarchical approach. If the latter is chosen, management of our sensing agents would involve a multi-level hierarchy of sensing, middle-manager, and overall manager roles.

The sets of role-goal-agent bindings and their parameters specify the long-term structure, role assignments, authority relationships, and communication paths of the designed organization. Although these long-term bindings are appropriate for many organizational goals, they are insufficient to satisfy organizational goals which may be better satisfied by establishing teams [4, 15, 13, 14, 8, 1]. Teams, coalitions, and congregations are temporary structures that are formed as needed to satisfy particular tasks when they enter the environment and are disbanded when the tasks leave the environment or are completed. In our simple example, tracking a newly detected vehicle might be done by creating a team whose membership changes as the vehicle moves through the monitored area. Teams are not strictly part of the organizational structure since the assignment of agents to roles

associated with the team will not be as long lived as the assignment of agents to roles to satisfy organizational goals. However, a team is not a purely operational construct either, since 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 have an effect on how it satisfies its other roles. Therefore, the organizational structure must account for and be prepared for team activity by its members.

We do not generate transient teams in our organization-design process, but we must ensure that the organizational structures and resources exist to generate effective teams operationally as needed. Thus, in our design process we reserve resources within agents capable of participating in teams. For the DSN example, this means finding role-goal-agent bindings for the leaf goals of TRACK, but setting the $team$ flag to true to indicate that the agent participates in the team only as needed. A team role is similar to an organizational role in that the agent with a team-role responsibility will have an expected number and frequency of messages to send and amount of work to do. The difference is that the agents bound to these roles will only be expected to perform those activities if and when they are called upon to join a team. Furthermore, 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.

Figure 4(c) shows an example set of bindings for a single agent participating in the DSN. For each binding, it specifies which organizational subgoal it is bound to, and the role and agents in that role to which it sends information. If the role is a teaming assignment such as FUSER→FUSE, it is signified with a superscript T.

2.4 Search and Suitability

In general, multiple roles can satisfy the same organizational subgoal, many agents can be bound to the same role-goal binding, and a single agent can play multiple roles simultaneously, making it computationally infeasible to generate all possible bindings. Therefore, we have developed a prototype system that uses organization-design knowledge and heuristics to generate a reasonable set of bindings. For the domain-specific portions of the design process, the heuristics rely on information provided by the developer in the quality, requirements, and decomposition specifications of the roles plus the capabilities of the agents. 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 similar process of finding role-goal-agent bindings for the coordination goals. The main difference is that the roles available for satisfying the coordination goals and the search heuristics exist within the organizational-coordination library as domain-independent knowledge. In the current prototype the parameters on coordination roles and goals are not fully generalized; some parameterization values still refer to problem-domain parameters. In future research, we plan to develop generic abstractions of

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

(a) Parameters and to-be-assigned list of SCAN goal

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

(b) Problem-Domain 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 represents one of the goals in Figure 3. A is the set of agents.

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
...

(c) Set of problem-domain and coordination role-goal-agent bindings for a single agent

Figure 4. Representation of goals, roles, and role-goal-agent bindings

problem-domain parameters (that would be included as part of the problem-specific knowledge) that would provide a completely clean separation of problem-domain and coordination parameters.

Although the heuristics above should lead to an organization that meets the performance requirements, they do not give enough information to rank a set of feasible candidate organizations all of which satisfy the requirements. We must consider other factors in how we evaluate them. For that it is important to have an organizational evaluation function that is based on user specified criteria to determine the utility of a particular candidate. In future work, we plan to develop a detailed evaluation capability both to evaluate fully specified organizations and to prune the search through partially complete bindings. For now, we rely on simple utility criteria stemming from the relative costs of agent load and communication.

3 Example Organization Designs

We present below four example organizational designs generated by our automated system on the goal tree and communication graph in Figure 3, the parameters in Figure 2, and the roles in Figure 4(b). We varied the input along several dimensions: size of the area to be scanned and number of agents available, the value of the acceptable track delay performance requirement, 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. Figure 5 summarizes the results.

In the first design scenario, we used 36 agents in a $90' \times 90'$ rectangular area with an acceptable track delay performance requirement of 3 seconds, and the cost of communication greater than that of agent loading. The resulting organization was a single-level hierarchy with 6 managers each managing 6 agents. The managers coordinated among themselves using a peer-to-peer mechanism. Furthermore, in order to minimize communication, there were 6 verifying and handling roles each multiplexed within the same agents as the managing roles. This organization corresponds closely to the hand-crafted organizational structure used for the EW Challenge Problem [7] where communication cost was a major concern. The performance of this organizational form relative to others was recently tested experimentally [5, 6]. 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.

When we switched the relative costs of communication and load, the resulting organizational design was still a single-level hierarchy, but the verifying and handling roles were no longer multiplexed within the same agents as the manager roles. Instead they were distributed to separate agents in order 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.

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

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

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

Agents	Area	Delay	Com. Cost	Load Cost
100	$150' \times 150'$	$3s$	0.6	0.4
Two-level hierarchy: 9 mid-level Managers. Verifi er 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 5. Example Organizational Designs

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. 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 this time with 9 managers and 2 upper-level managers which coordinate using a peer-to-peer mechanism.

Overall, we were quite pleased that our design system produced such different organizational forms given the changes to the environmental characteristics and performance requirements we presented it with. These results

confirm for us the usefulness of our approach in 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 for the field of multi-agent organizational design. It relies on a separation between the problem-domain and the organizational coordination-domain to generalize coordination mechanisms across domains, requiring a developer only to supply problem-specific 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 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 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, we must 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 coordination-domain knowledge so as to parameterize the coordination roles more appropriately. Part of this will involve understanding the distinguishing features of 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 goals interrelate.

References

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

[2] D. D. Corkill. *A Framework for Organizational Self-Design in Distributed Problem-Solving Networks*. PhD thesis, University of Massachusetts, Amherst, Massachusetts 01003, Feb. 1983. (Also published as Technical Report 82-33, Department of Computer and Information Science, University of Massachusetts, Amherst, Massachusetts 01003, December 1982.).

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

[4] M. S. Fox, M. Barbuceanu, M. Gruninger, and J. Lin. An organization ontology for enterprise modelling. In M. Prietula, K. Carley, and L. Gasser, editors, *Simulating Organizations: Computational Models of Institutions and Groups*, pages 131–152. AAAI/MIT Press, 1998.

[5] B. Horling, R. Mailler, and V. Lesser. A Case Study of Organizational Effects in a Distributed Sensor Network. Computer Science Technical Report 04-03, University of Massachusetts, January 2004.

[6] B. Horling, R. Mailler, M. Sims, and V. Lesser. Using and Maintaining Organization in a Large-Scale Distributed Sensor Network. *Proceedings of the Workshop on Autonomy, Delegation, and Control (AAMAS03)*, July 2003.

[7] B. Horling, R. Vincent, R. Mailler, J. Shen, R. Becker, K. Rawlins, and V. Lesser. Distributed sensor network for real time tracking. In *Proceedings of the 5th International Conference on Autonomous Agents*, pages 417–424, Montreal, June 2001. ACM Press.

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

[9] 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. Zhang. Evolution of the GPGP/TAEMS Domain-Independent Coordination Framework. *Autonomous Agents and Multi-Agent Systems*, 9(1):87–143, July 2004.

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

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

[12] H. E. Pattison, D. D. Corkill, and V. R. Lesser. Instantiating descriptions of organizational structures. In M. N. Huhns, editor, *Distributed Artifi cial Intelligence*, Research Notes in Artifi cial Intelligence, chapter 3, pages 59–96. Pitman, 1987.

[13] T. Sandholm and V. Lesser. Coalitions among computationally bounded agents. *Artifi cial Intelligence, Special Issue on Economic Principles of Multi-Agent Systems*, 94(1):99–137, Jan. 1997.

[14] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *Artifi cial Intelligence*, 101(1–2):165–200, 1998.

[15] 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. *Artifi cial Intelligence*, 110:215–240, 1999.