Reflections on the Nature of Multi-Agent Coordination and Its Implications for an Agent Architecture

by
V.R. Lesser

CMPSCI Technical Report 98-10
August, 1998

Appeared in *Autonomous Agents and Multi-Agent Systems*, Kluwer Academic Publishers, 1, 89–111, July 1998.

Reflections on the Nature of Multi-Agent Coordination and Its Implications for an Agent Architecture *

by
Victor R. Lesser

CMPSCI Technical Report 98-10
February 1998

Abstract

The development of enabling infrastructure for the next generation of multi-agent systems consisting of large numbers of agents and operating in open environments is one of the key challenges for the multi-agent community. Current infrastructure support does not materially assist in the development of sophisticated agent coordination strategies. It is the need for and the development of such a high-level support structure that will be the focus of this paper. A domain-independent (generic) agent architecture is proposed that wraps around an agent's problem-solving component in order to make problem-solving responsive to real-time constraints, available network resources and the need to coordinate — both in the large and small, with problem-solving activities of other agents. This architecture contains five components, *local agent scheduling*, *multi-agent coordination*, *organizational design*, *detection and diagnosis* and *on-line learning,* that are designed to interact so that a range of different situation-specific coordination strategies can be implemented and adapted as the situation evolves. The presentation of this architecture is followed by a more detailed discussion on the interaction among these components and the research questions that need to be answered to understand the appropriateness of this architecture for the next generation of multi-agent systems.

Key words: coordination, multi-agent architecture, agent societies

*This material is based upon work that has been sponsored by the Dept. of the Navy, Office of the Chief of Naval Research (under Grant No. N00014-95-1-1198), the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF (under agreement number F30602-97-1-0249), the National Science Foundation (under Grant No. IRI-9523419), and Network General Corporation (now part of Network Associates). The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research Projects Agency (DARPA), Air Force Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

1. Introduction

The concept of large ensembles of semi-autonomous, intelligent agents working together is emerging as an important model for building the next generation of sophisticated software applications. This model is especially appropriate for effectively exploiting the increasing availability of diverse, heterogeneous and distributed on-line information sources, and as a framework for building large, complex and robust distributed information processing systems. A prototype example of this model is WARREN, a recently developed multi-agent system [9, 10, 11, 61] for doing portfolio management on the Internet. The development of enabling infrastructure for mobile computing and interoperability among programs residing at distant sites, and new generations of distributed operating systems, has made and will continue to make the construction of systems based on this model much easier. However, this infrastructure support mainly deals with low-level protocols for transmission of information and does not embed any deep models about the nature of agent coordination. Simply defined, agent coordination involves the selection, ordering and communication of the results of agent activities so that an agent works effectively in a group setting. Agent coordination needs to take into account in its reasoning the quantitative character of interactions among agent activities [15], the performance characteristics and resource demands of activities, the likely future activities of agents, resource loadings, hard and soft deadlines for completion of agent activities [31], and also more symbolic information such as the beliefs, desires and intentions of agents [7, 30, 33, 35, 52] and the desirability (or undesirability) of certain joint states of agents being achieved [2, 60]. It is the development of a high-level framework that exploits such analytic and symbolic information that will be the focus of this paper[1].

Coordination of agent activities becomes necessary when there are interdependencies among agent activities, e.g., through contention for resources needed for their execution or due to these activities (subproblems) contributing to the solution of a larger problem. For example, a simple situation of subproblem interdependencies occurs when agents are intending to work on the same or overlapping subproblems and have alternative methods or data that can be used to generate a solution. Coordination in this case involves making decisions about whether to have both agents work on the subproblem concurrently or, if sequentially, then in what order; or if only one agent is to solve the subproblem, which one. Another form of interdependence occurs when two subproblems are part of a larger problem in which a solution to the larger problem requires that certain constraints exist among the solutions to its subproblems[2]. Included in this latter case is the simple situation where the results of one subproblem are needed to solve another, and the more complex situation where there are constraints among joint states of agents. Coordination in this second case involves making decisions about when each agent should begin to solve its subproblem and whether and when intermediate results of a subproblem solution are worthwhile transmitting. Additional interdependencies among subproblems, not inherent in the problem domain, arise when it is not possible to decompose the problem into a set of subproblems such that there is a perfect fit between the computational requirements for effectively solving each subproblem and the location of information, expertise, processing, and communication resources in the agent network [42, 43]. When there are these interdependencies, overall system performance is significantly affected by the choice of methods used to solve a subproblem, the order of solving the subproblems, the time at which a subproblem will be solved, and what aspects of subproblem solutions are transmitted and to whom.

The usefulness of coordination can be seen in the simple situation where one agent needs the results of a subproblem that another agent is solving. If it can be arranged that the producing agent will deliver in a timely fashion the desired result so that the consuming agent does not have to idle waiting for the results, then system performance is improved.

On the surface this coordination decision is simple. However, suppose that the producing agent has other tasks to do with their own deadlines in addition to producing a result for the other agent. To further complicate this decision process, the agent may have alternative methods for doing those tasks that trade off the quality of the task solution against the time to complete the task. Similarly, the consuming agent may also have flexibility about when it does its tasks because there are other tasks it is also working on, and it may also be able to make trade-offs in how it accomplishes its tasks. The coordination decision then involves a complicated optimization problem of ordering tasks and selecting how to accomplish them that spans multiple agents. In general, the importance of effective coordination and the corresponding need for more sophisticated coordination mechanisms increases in situations where there are complex subproblem interdependencies among agents, where there are time-pressures and resource bounds which preclude all goals of the system being solved in an optimal manner, where there are many choices available about how to solve a goal, and where the goals being solved and the agents and resources available to solve them are changing over time.

Negotiation, the process of arriving at a state that is mutually agreeable to a set of agents, is intimately related to coordination. The negotiation process can be used as part of a multi-agent coordination algorithm that implements, for instance, a contracting mechanism for getting one agent to commit to solving a subproblem for another agent. It may also be used by agents to ensure that their solutions to subproblems are compatible. In this case, this domain-level negotiation process places constraints on the order of agent activities in solving the specific subproblems and requirements for the transmission of intermediate results that must be adhered to by the coordination algorithm.

Effective coordination strategies will be very important for agents in next-generation multi-agent systems. These agents will need to be highly adaptive due to their "open" operating environments where the configuration and capabilities of other agents and network resources could change dynamically. One of the ways that such agents can be adaptive is to have multiple ways of solving their subproblems so they can adjust their problem solving to produce the best possible result given their available processing, communication and information resources. Agents can also be more adaptive if they are not restricted to solving one goal at a time but are able to flexibly interleave their activities to solve multiple goals concurrently. Similarly, an agent should be able to choose to carry out an entire task itself, or it could elect to coordinate with appropriate agents to perform tasks collectively. Additionally, systems consisting of hundreds to thousands of such agents will need to be able to form and evolve higher order social structures such as teams and organizations to exploit collective efficiencies and to manage emerging situations [27]. It is this ability to adapt intra-agent and inter-agent problem solving to the dynamics of the environment in both short- and long-term ways that will differentiate these types of agent-based systems from more conventional distributed systems architectures where adaptability, especially at the domain problem-solving level, is not a primary motivation [63].

The fundamental issue to be addressed in this paper is: what are the basic functions and interaction patterns necessary for an agent architecture to support the construction of such systems and to allow them to operate efficiently and robustly? The answer to this question will of necessity be speculative since there is no substantial experience in building multi-agent systems of this anticipated scale and complexity. It will be a personal view based on my experiences over the last 25 years in thinking about and building a wide range of multi-agent systems for applications such as distributed situation assessment [3, 22, 41, 44, 57], distributed scheduling and resource allocation [7, 45, 48, 55], multi-expert design [38], concurrent engineering [27] and cooperative information gathering [49]. These systems have all involved agents performing significant local computation, and often involve agents that cannot fully or accurately complete their local processing without interacting with other

agents [42, 43]. In these systems, generating a solution to a problem is often not a straightforward assembly process based on combining the solutions of agents' local subproblems. Rather, this process can be a multi-step and incremental process involving a dialogue among agents using information at multiple levels of abstraction [3, 4]. Each step of the dialogue may require the agent to reason about the new information in relation to its existing knowledge.

Another characteristic of these systems is that they have attacked combinatorially explosive problems where it is impractical to explore all possible solutions. Additionally, they are often implicitly or explicitly part of a larger system, i.e., they are subsystems; thus, their performance criteria must be responsive to the current needs of their client. For this reason, real-time performance and satisficing behavior needs to be an integral aspect of any agent that operates in a large context. These needs for real-time performance and satisficing behavior combined with the potentially complex nature of agent interaction justifies the need for sophisticated coordination strategies in certain situations. There are numerous examples occurring in even small agent systems where inappropriate coordination has led to the system generating suboptimal answers or no answer at all, not meeting deadlines and/or wasting considerable resources on inappropriate activities.

An early example of this problem occurred in the distributed Hearsay-II system [41], which was one of the first multi-agent systems constructed. In this system of three agents which were interpreting overlapping fragments of speech data, the lack of effective coordination led to agent problem solving being distracted by unreliable information (partial and tentative solutions to local subproblems) that was received from another agent. This distraction resulted in agents performing unnecessary work and delaying the generation of a correct solution because of computational resources used in pursuing the implications of the incorrect information that it received from another node. Other manifestations of ineffective coordination in this system were agents rederiving results already produced by another agent and the transmission of information among agents that was no longer timely or that did not contribute to the current line of reasoning of the receiving agent. These problems of ineffective coordination can be traced to agents having a very limited view of the state of network problem solving; it was based solely on the results they had received from other agents — they had no way to compare their state of progress in problem solving with other agents; they had no knowledge of what subproblems other agents had worked on, were working on or intended to work on in the near future, nor the progress or lack of progress these agents had made on solving these subproblems [43]. Without agents having some perspective on the activities of agents solving interrelated subproblems, there will of necessity be some level of incoherence among agents. As will be discussed in the next section, in some situations a certain level of incoherence is acceptable given the costs of acquiring and processing information to eliminate this incoherence.

The approach that will be suggested (based on the multi-agent systems that I have constructed) for addressing the question of what is an appropriate multi-agent architecture starts with the basic premise that an agent's problem-solving component is sophisticated — possessing a certain level of self-awareness about its current and intended goals including the current state of their achievement and the performance characteristics of the approaches that are possible to solve them, and being able to flexibly adapt its problem solving to the available resources and current criteria for goal satisfaction. The ability of an agent's problem-solving component to provide abstracted information about its state permits the implementation of complex coordination strategies and, in turn, the ability of an agent to adapt its problem solving permits a coordination strategy to align an agent's activities to satisfy the more global objectives of the system. Implicit in this discussion is that the appropriate granularity of agents' computation is medium- to large-grain. It is in the context of this type of computational granularity where problem solving for effectively interacting

with other agents could be quite complex, yet not overwhelm the agent's domain problem solving.

Another important premise of the proposed approach is that the distinction between self-interested (competitive) agents that are trying to optimize their own local performance and cooperative (benevolent) agents which are trying to optimize overall system performance is important but not an overriding factor in the design of coordination mechanisms for the complex agent societies that operate in open environments. In fact, I feel agents that populate such societies will use performance criteria that combine both local and non-local perspectives and that these performance criteria, in terms of the balance between local and non-local performance objectives, will change based on emerging conditions. Thus, I see this distinction between self-interested and cooperative agents blurring in the next generation of large and complex multi-agent systems. The basis for this view is that agents that operate in these complex societies and open environments will have to cope with a tremendous amount of uncertainty, due to limited computational and communication resources, about how to best perform their local activities and how their activities will affect the agent society more broadly — there will be complex interactions among the activities of agents, e.g., chains of interdependencies among agents' subproblems. These factors will lead to self-interested agents behaving in more cooperative ways so that they can acquire useful information from other agents and help other agents in ways which will eventually improve their own local performance[3]; in turn, cooperative agents will behave in more self-interested ways, given the costs of understanding the more global ramifications of their actions, as a way of optimizing overall performance of the society[4]. The only real difference between self-interested and cooperative agents is the evaluation function that they use to rate the importance of performing specific activities. As will be discussed more fully in the next section, by using a coordination framework that at its lowest level makes decisions from a quantitative perspective, a range of higher level coordination mechanisms can be constructed within this framework that support both self-interested and cooperative agents.

There is much valuable insight to be gained at this stage of the field's development by beginning to consider what a comprehensive multi-agent system architecture would look like. Frameworks have been developed [2, 6, 23, 35, 36, 52, 56, 62] that are good initial starts but each makes simplified assumptions about the complexity of cooperative behavior and agent capabilities and does not directly deal with one or more of the following issues: large-scale agent societies, adaptation to resource availability, hard and soft real-time deadlines, trade-offs between resource usage and quality of solution, complex interdependencies among agents' activities and domain-independence. The remainder of the paper will lay out my view of some of the important concepts that need to be included in a multi-agent architecture followed by my proposal for one such architecture.

2. Important Architectural Concepts

A central feature of large agent societies is that they will be populated by heterogeneous agents created by many designers. The question naturally arises as to how much effort designers must expend to adapt their agents to effectively interact with other agents; that is, what aspects of the basic agent architecture are generic or domain-independent? One approach is to provide a set of operations and associated protocols for locating and communicating with agents, such as KQML [24]. However, in this approach, which is knowledge "poor," the decision about when to use a protocol, what information to transmit, what order to execute tasks, etc., is left to the designer. Instead, a high-level coordination framework that can be built on top of such protocols will be proposed. This framework, which wraps around the agent's problem-solving component, will automatically make all the coordination decisions, once an agent has described in an

abstracted form its needs and capabilities for interacting with other agents in a domain-independent way. The type of information that needs to be provided includes an agent's goals and criteria for their successful performance, the performance characteristics and resource requirements of the alternative methods it possesses for accomplishing its goals, and both the qualitative and quantitative interdependencies among its methods and those of other agents [12, 15].

This high-level approach minimizes the amount of domain-dependent code and knowledge that the designer must specify to achieve effective coordination with other agents in a variety of situations, and permits the designer to build agents without significant expertise in the development of coordination strategies. However, this approach does have the drawback of forcing the designer to add code to the agent's problem-solving component, which will be referred to here as the *task assessor module*, to present the necessary information in the appropriate form to the coordination framework. The better the task assessor is at predicting the future goals of the agent and the performance characteristics of the methods available to solve the goal, the more effective the framework is in making coordination decisions; it also introduces a basic overhead that all agents must pay when using this domain-independent approach.

I am not advocating that this framework be closed. It is crucial that the designer be able to introduce, if desired, domain-specific higher level coordination strategies via a programming language interface that builds on framework capabilities. An example of where this would be appropriate is where an agent designer wants to build a complex domain-level negotiation protocol among agents. This protocol makes decisions about what to transmit and what computation to perform based on the responses of the other agent and the local state of the agent. The domain-independent framework uses abstracted information about the activities of agents so that it will be applicable to a wide variety of agents, and its decision process will not be overwhelmed by extraneous details. However, this abstracting can remove domain-specific knowledge (such as information about some aspect of the joint state of two agents) that may be usefully employed in making coordination decisions. The focus on keeping the high-level framework open allows the agent designer to have some way of telling the high-level framework either directly about this missing knowledge, or indirectly, by describing the implications of this knowledge on coordination decisions. In this way, it is hoped that a unified approach to agent coordination that includes both quantitative and symbolic reasoning can be constructed to fit the needs of the particular application.

In this architecture, coordination decisions are made on three separate and semi-autonomous layers that operate concurrently: an agent organization layer, a small agent group coordination layer[5], and a local agent scheduling layer. Each layer makes decisions, respectively, that involve different durations and specificity. The organization layer makes decisions that are of long-term duration and the least specific, while the local agent scheduling layer makes decisions that are of short-term duration and very detailed. The three control layers interact by higher layers providing constraints (policies) to lower levels that modulate (circumscribe) their control decisions. These constraints indicate, for example, what level of resources should be used to make control decisions (e.g., how much effort should the local scheduler spend in finding the best sequence of tasks to execute) and to determine the scope of the decisions (e.g., which tasks in which agents need to be coordinated, and what type of coordination is necessary) [8, 16]. The interaction among layers is not just top-down but also bottom-up. As constraints flow down the layers, information that flows in the other direction allows these constraints to be modified in case they cannot be met or lead to inappropriate behavior. This type of interaction provides the capability for control layers to negotiate to reconcile their different perspectives on what actions to take in the current situation [25]. This architecture should make no

specific decision about which agents or set of agents implement each control layer. Rather, the agent organization for control should, if it is warranted, be separable from the agent organization for domain problem solving [8, 21, 22, 37, 43].

This perspective on coordination is closely associated with the viewpoint that both long- and short-term coordination can be specified in terms of commitments that have varying duration and specificity [20]. The ability to appropriately bound the intentions of agents, and to create and sufficiently guarantee the commitments of agents to accomplish certain tasks, is at the heart of efficient, organized behavior [5, 16, 20, 26, 29, 33, 34]. Another way of seeing long-term commitments is that they define agent roles and responsibilities. For large agent societies, organizing agents in terms of roles and responsibilities can significantly decrease the computational burden of coordinating their activities by limiting both the information that needs to be acquired and the scope of the decision process. However, these assignments should not be so strict that an agent does not have sufficient latitude to respond to unexpected circumstances, nor should they necessarily be fixed for the duration of problem solving [8]. An example of the efficacy of assigning long-term roles is the following: instead of an agent incurring the computational overhead of dynamically coordinating with a group of agents to determine which agent is most appropriate to provide a desired result and at what time, the organizational design could have prescribed ahead of time that a specific agent take on the role of providing the agent with the desired result by a certain time. This role assignment could have been based on an analysis of which tasks the agents would likely perform, their relative importance, frequency, deadlines and performance characteristics, etc. To the degree that these characteristics are not highly variable, it may not be efficient from an overall system perspective to try to dynamically optimize which agent is the most appropriate in the current situation to provide the needed information.

I feel that it is crucial when dealing with large agent societies to be able to reason about the assumptions behind the organizational design in order to effectively adapt it when the operating environment changes. The organizational design encodes knowledge about when, where and which type of coordination strategies are useful in the current environment. To the degree that the system can either re-derive or explicitly represent the assumptions behind these design decisions, the system can more effectively detect and diagnose the causes of inappropriate or unexpected agent behavior. The results of this analysis will let the system adapt the agent organization to correct for this problem based on a revised set of assumptions about environment and agent performance characteristics [58, 59]. Consider the organizational design discussed above where one agent is assigned the role of providing a specific result at a defined time to another agent based on assumptions about what tasks it will likely work on and their computational requirements. Suppose the agent that is supposed to receive the result does not get it at the expected time. A diagnosis process could then be invoked to find out why. Suppose that it found out that the agent was delayed in producing the desired result because the associated computation was taking longer than expected due to the saturation of the database server that was used by the computation. This understanding could be directed back to the organizational design component which would revise its assumptions about task and resource characteristics; the analysis of these updated assumptions could result in organizational changes such as revising the due date on the commitment to a later time, finding an alternative way of generating the result that does not need the saturated resource, finding a way of unsaturating the resource, etc.

Another important idea is that the use of resources to generate optimal coordination patterns is not justified in all situations. There is no one best approach to organizing and controlling computational activities for all situations when the computational and resource costs of this control reasoning is taken into account. Instead, each agent should be able to tailor, to the

specifics of the current situation, *all* aspects of control reasoning to balance the resource requirements of this reasoning against the gains achieved by more coherent agent behavior [13, 14, 20, 21, 43, 46]. An associated corollary is that the system needs a rich set of criteria (performance objectives) that can be used to define the characteristics of the satisficing behavior that is most appropriate for the given situation [66, 67]. Thus, even though it has been argued that sophisticated coordination is needed in certain situations, it is also the case that the use of resources for implementing sophisticated coordination is counter-productive in other situations. Therefore, any architecture that will be appropriate for a wide class of multi-agent systems must be able to support a range of coordination strategies in terms of their sophistication and the information they require.

I also feel strongly that the right way to view the decision about how best to coordinate can be framed as a complex, discrete optimization problem based on how coordination actions, in a quantifiable way, contribute to high-level system tasks meeting their performance objectives and the relative importance of each of these tasks. Efficient and effective coordination must account for the benefits and the costs of coordination in the current situation in a quantitative way. The current situation includes the goals (and their importance or value) that the agent is currently pursuing and likely to pursue in the near term, the performance characteristics of the methods available to the agent for achieving its goals, the requirements these goals/methods impose on other agents, the requirements that the goals/methods of other agents impose on this agent, the state of network resources and domain constraints on agent activities. Purely symbolic reasoning about costs and benefits can be extremely complex, particularly in large systems and open environments, or where agents can simultaneously pursue multiple goals. Thus, the ability to reason quantitatively about these benefits and costs of coordination seems essential for effective system operation where there is a large set of situations that need to be reasoned about [16, 43]. Another way of saying this is that there must be a mechanism at the lowest control layer to arbitrate among the various objectives of the agent, such as the local processing goals it is pursuing and the various requests by other agents for its assistance. It is this view that leads to what I feel is a principled and general approach to coordination even though heuristic methods will be needed to get approximate solutions to this NP-hard optimization problem.

As discussed in the introduction, the ability to consider the worth of alternative courses of action from a quantitative perspective, and to allow the evaluation function to be agent-specific, permits the development within this framework of coordination strategies that support both self-interested and cooperative agents. For example, if one agent needs another agent to perform a particular task by a particular time, but the other agent is self-interested and the task has little value to this agent, the value associated with the satisfaction of the commitment made with the other agent will be very slight or even non-existent; the self-interested agent is accordingly unlikely to perform the given task as requested unless value is provided. The realization that the value associated with completing a task can be a complex calculation that is agent-dependent and can take into account such things as the other tasks the agent can potentially perform, the importance attributed to completion of this task with a certain quality and by a certain deadline, etc., leads us to an understanding of how negotiation is incorporated into the quantitative coordination model. A new coordination mechanism could be created to coordinate activities between not-fully-cooperative agents, and this mechanism would determine the effect that coordinating activities has on the local value of particular tasks. For example, if two agents negotiate and strike a deal for one agent to perform a task for another agent, and the agent is receiving a high fee for performing the task, then a high value will be associated with the task and the agent will consider it accordingly against other local tasks that need to be done.

The next section details a proposed multi-agent architecture based on these ideas.

3. A Proposal for a Multi-Agent Architecture

The proposed architecture builds upon the Generic Partial Global Planning (GPGP) framework for coordination of small teams of agents [16, 39][6]. The basic idea behind GPGP is that each agent constructs its own local view of the activities (task structures) that it intends to pursue in the short-to-medium-term time frame, and the relationships among these activities. This view can be augmented by information from other agents, thus becoming a view that is not entirely local (*ergo*, Partially Global). Individual *coordination mechanisms* that are part of GPGP help to construct these partial views, and to recognize and respond to particular task structure relationships by making commitments to other agents. These commitments result in more coherent, coordinated behavior by specifying the affects of other agents' actions on the tasks an agent will execute, when they will be executed, and where their results will be transmitted.

An example of a mechanism that reduces redundant activity is one that first determines whether agents are intending to perform the same activities (based on the detection of an overlapping task relationship and multiple intentions to perform the activity denoted by the action being on the scheduling queue of multiple agents). Once this situation is recognized, a dialogue is initiated among the agents. Analysis of the current performance criteria and the scheduling flexibility of agents results in a decision to have one or more of the agents perform the activity and the other agents remove the activity from their schedules. This decision will result in the introduction of commitments to send the results of the activity to the other agents that had intended to perform the activity. If the agents that expect to receive the results either do not receive them in a timely fashion (or the quality of the result is not as expected) or are warned ahead of time that this is likely to occur based on monitoring of task execution, they will then reschedule their local activities so that the desired activity is performed. Integral to GPGP is a domain-independent scheduler [67] that, based on commitments, agents' goals, agent activity constraints and the current performance criteria, creates a schedule of activities for the agent to perform that tries to optimize agent performance. This scheduling process is by nature heuristic due to the inherent combinatorics of the problem [67]. GPGP coordinates the activity of agents through modulation of their local control as a result of placing constraints and commitments on the local scheduling process. No single coordination strategy will be appropriate for all task environments, but by selecting from a set of possible coordination mechanisms (each of which may be further parameterized), a wide set of different coordination responses can be created.

The central representation that drives all of the GPGP coordination mechanisms is that of the local (and non-local) task structures. Several important pieces of information are captured in the task structure. These include: (a) the top-level goals/objectives/abstract-tasks that an agent intends to achieve or can be requested by another agent to achieve; (b) one or more of the possible ways that they could be achieved (expressed as an abstraction hierarchy whose leaves are basic action instantiations, called *methods*, that are characterized statistically via discrete probability distributions in three dimensions: quality, cost, and duration); (c) a precise, quantitative definition of the value of a task or method in terms of how it affects agent objectives based on measurable characteristics such as solution quality, cost and time; (d) task-task relationships that indicate how basic actions or abstract task achievement affect tasks elsewhere in the task structure[7]; and e) a task-resource relationship that indicates how the execution of a task affects the state of the resource which it uses during execution, and how the state of the resource affects the performance characteristics of the tasks that use it[8]. Optionally, organizational information can also be specified about which agents should be asked to achieve certain non-local task relationships and how much effort should be spent in trying to coordinate with other agents so as to achieve a non-local

task relationship. The language used to represent these task structures is called TÆMS [12, 15, 66], and can be thought as a pre-compiled network of possible plans for achieving a desired goal. A simple TÆMS task structure is shown in Figure 1. Unlike many other approaches, TÆMS deals with *worth-oriented* [53] environments where a goal is neither black nor white, but rather has a degree of achievement associated with it. It represents this type of environment by tracking a quantitative vector of task characteristics or criteria over which some utility preference may be expressed [66]. TÆMS also allows many task structures to be active at once, representing several objectives — all of which must be achieved to some degree. The agent's task structure view may change over time due to uncertainty or a dynamically changing environment. Some of that contingency can be represented in TÆMS by having different outcomes associated with task completion as shown in the "Find Information on WP Products" task in Figure 1; however, more completely representing those contingencies, in most cases, is not cost effective in terms of both the costs incurred by the agent generating the more complex representation and the coordination framework processing this representation. A balance needs to found between contingency representation within TÆMS and the dynamic updating of the TÆMS structure by the task assessor module when the current task structure is no longer representative of agent problem solving; in the latter case, the coordination framework cannot reason about contingencies in making its decision. Furthermore, a highly reactive or opportunistic agent should not generate an extensive representation of its activities since this additional information would not be valuable for coordinating the activities of this type of agent, and in fact, would be counterproductive in terms of wasted computational resources. More generally, the agent, by choosing the level of abstraction and the temporal scope of its representation of computational activities, defines the level of detail at which its activities will be controlled and the type of coordinated interactions that are possible with other agents.

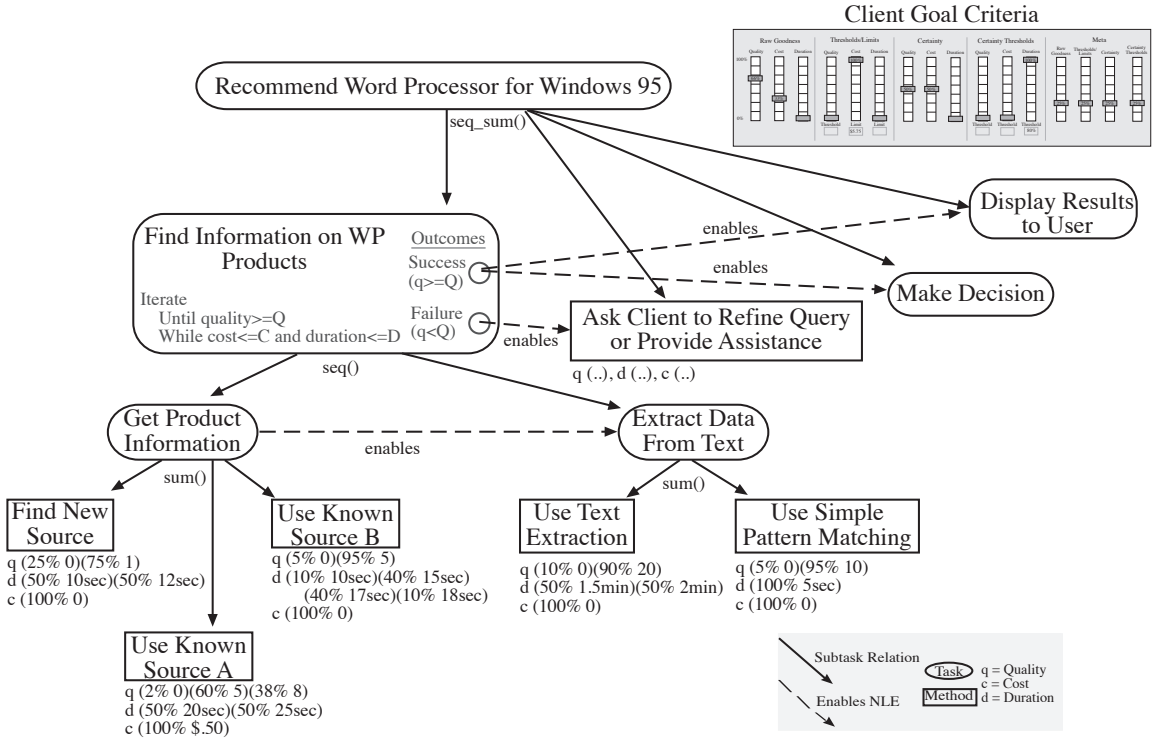

Figure 1: An example of a TÆMS task structure for an Information Gathering Agent

The results of applying GPGP to a number of different applications, including distributed situation assessment, information gathering and management, coordination of concurrent

engineering activities, and hospital scheduling [18, 19, 47, 49, 50] have led me to believe that the basic approach to small agent team coordination, based on quantitative coordination relationships represented in TÆMS and the generation of commitments among agents that then constrain local agent scheduling, is a very powerful and general framework for implementing coordination mechanisms. Further, GPGP can be naturally extended to be highly situation-specific, where the overhead for coordination can be adjusted for the specific coordination problem. This can be accomplished by substituting dynamically acquired knowledge and dynamically generated commitments for a range of prior knowledge[9]. Additionally, GPGP can be easily extended to allow the implementation of more top-down and contracting types of coordination mechanisms [39]. It is these extensions to GPGP which will be built upon to create a multi-layered control architecture in which an organizational design defines situation-specific policies for multi-agent coordination.

The multi-agent architecture, pictured in Figure 2, is based on each agent having some of the following components: *local agent scheduling*, *multi-agent coordination*, *organizational design*, *detection and diagnosis* of inappropriate behavior in the face of environmental change, and *on-line learning* of knowledge that can improve the performance of the other components. The last three components represent extensions to the GPGP architecture, added to create an architecture more applicable to large agent organizations. All the components will use as their basic representation an extended version of TÆMS that includes statistical information about long-term environmental conditions such as resource availability and task arrival patterns.

The upper section of Figure 2 contains components that agents use to execute their short-term coordination behaviors. The lower section contains the additional components that agents will need to alter their long-term behavior. These latter components are involved in organizational design and not all agents will need to have these components. Agents that are performing domain problem solving must contain the local agent scheduling and multi-agent coordination components. Agents that are not doing domain problem solving in turn do not require these two components. Other than these constraints, the choice of an agent's components provides a way of creating agent organizations that have different performance characteristics in terms of the responsiveness to changing environmental conditions, the effectiveness of agent re-organization based on the level of knowledge that can be brought to bear in adapting the organization, and the survivability characteristics of the organization. The achievement of these characteristics has to be balanced against the computational and communication overhead incurred by more agents in the organization being configured with components for organization adaptation.

Component Interactions

In order to understand how the proposed multi-agent architecture functions, the details will be presented on how the components of this architecture interact so that agents can respond to both long- and short-term coordination policies. These policies can be adapted as the result of the actions of agents and the status of resources. In thinking about these interactions, it is important to keep in mind the optimization perspective on coordination. The purpose of these component interactions is to limit the amount of knowledge that needs to be dynamically acquired and processed in order for the set of local agent optimization problems to be an "adequate" reflection of the global optimization problem. When these local agent optimization problems are no longer an adequate reflection then it is important to recognize this situation and reassess how these local problems can be modified to more closely mirror the global optimization problem.

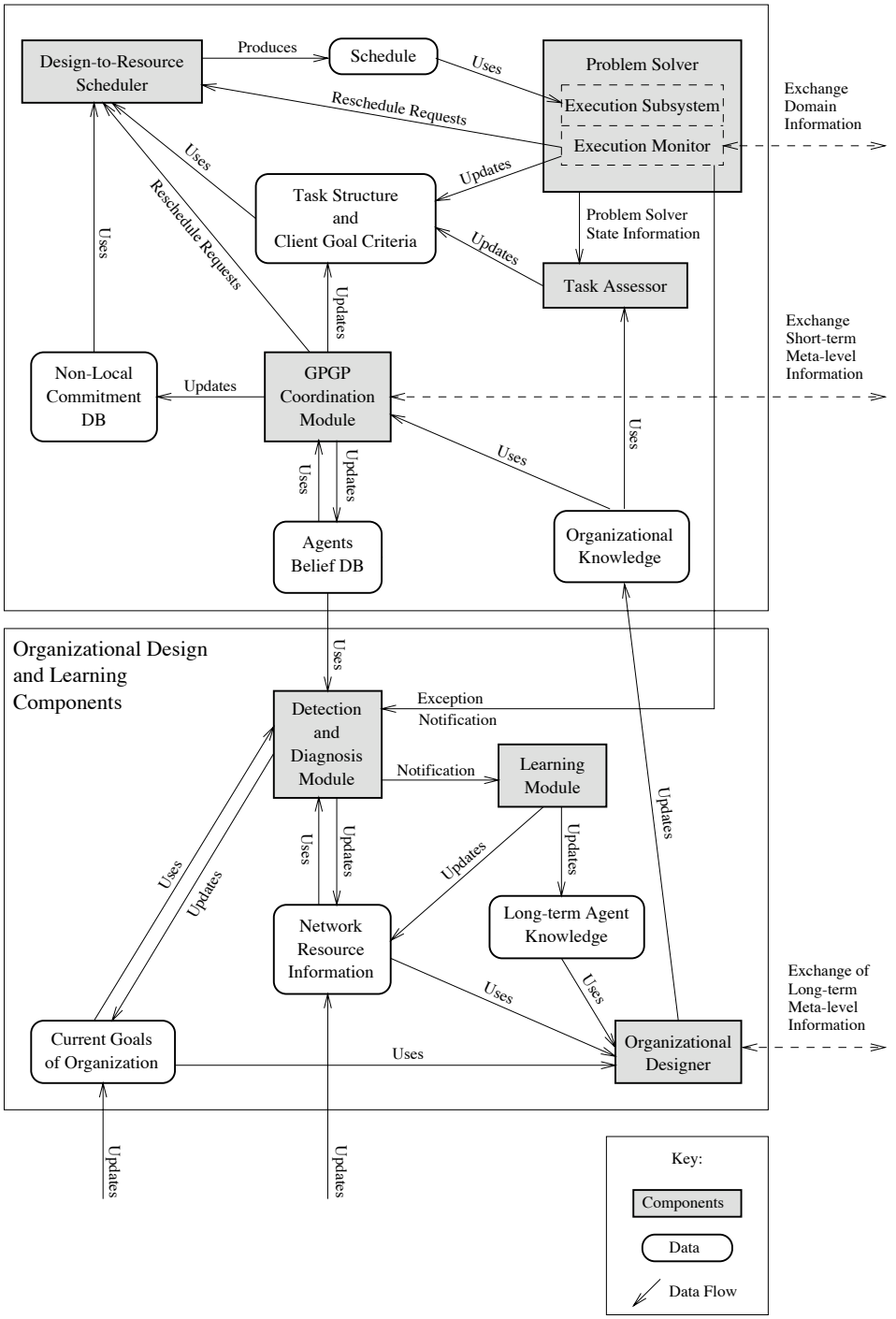

Figure 2: An Agent Architecture for a Large-Scale Multi-Agent Environment

From a top-down perspective, the *organizational design components* in one or more agents work collaboratively to design an organization appropriate for those agents and the resources they are responsible for managing. The inputs to organizational design are: 1) the current performance goals of the system; 2) long-term agent knowledge about typical tasks together with their frequencies of occurrence, possible relationships among other tasks in terms of co-occurrence, and specific performance goals and resource requirements; and 3) the available resources and their performance characteristics including the expertise

(processing capabilities, know-how) of agents. The output of this process is an organizational design — a description for each agent of: 1) tasks it is responsible for handling and their importance; 2) coordination relationships among tasks handled by other agents to be considered when scheduling this task, and the level of effort to be spent in establishing them; and 3) long-term commitments to be honored, their priority, and which agents should be notified when these commitments cannot be honored.

An organizational design is used when an agent receives a request to execute a new goal (with its associated task structure). This request can come from either its local problem-solving component or other agents. A part of the local problem-solving component is a *task assessor* module that is responsible for determining the problem-solver's expected near- to medium-term activity patterns for achieving a specific goal and translating these patterns into a TÆMS task structure[10]. Based on the organizational design, an agent's *coordination component* decides both how to constrain the characteristics of the task so the agent can meet coordination requirements with other agents and how to value the task's importance. For example, the coordination component might introduce a deadline if the task's results are useful to another agent or to associate a high value (relative to the importance attributed to the agent's own tasks) to a priority task request from an agent who is playing an important role in the current organization. Such decisions are made in collaboration with the coordination component of relevant agents.

The new task, with any constraints and related commitments that have been added by the coordination component, is then passed to the *scheduling component*. It schedules this task and its associated subtasks in the context of possible other tasks that the system is currently working on. The result of this scheduling is an ordered list of subtasks to be accomplished with associated performance envelopes. The local problem-solving component then executes the subtasks in the specified order and reports their execution status to the scheduling component. The scheduling component then updates its database to reflect the current state of task processing. Additional coordination and scheduling can occur during task execution and after the execution of any subtask, so that the system responds to changing circumstances such as the arrival of new high-level tasks or the failure of one or more subtasks.

This top-down coordination process can be modified in a number of ways. At the lowest level, the *problem-solving component*, as a result of a partially executed subtask, can recognize that the subtask will not fall within the performance envelope indicated by the scheduler. This will cause the scheduler to determine whether there exists an alternative way of completing the execution of the high-level task to meet the specified performance goals and commitments associated with its successful completion. The inability to meet these goals and commitments will result in either the local problem-solving component or another agent being notified to take corrective action.

The problem solver, as a result of executing a subtask, may also realize that the high-level task description specified by the task assessor is no longer representative of the problem solver's activities. This will result in the local problem solver indicating to the scheduler that the high-level task cannot be completed successfully, or the task assessor providing a modified description of how to complete the high-level task in terms of either updated performance characteristics of subtasks or changes in the type of subtasks and their relationships that will be required to complete the high-level task. Rescheduling will then occur with the possibility of also reinvoking the coordination module to establish revised commitments for this task.

The scheduler can also cause the reinvocation of the coordination module during the initial stages of scheduling because tentative commitments that the coordination module has

established cannot be met. Additionally, the scheduler is pro-active in that it can also suggest ways of partially meeting these commitments. This information can be used by the coordination component as a basis for negotiating with other agents' coordination components to find a compromise set of commitments. The scheduler can also suggest to the coordination component that it attempt to determine whether there is another agent willing to execute some task in a timely fashion rather than it being scheduled locally.

Agent activities and resource utilization characteristics are constantly monitored for situations that may indicate that the organization is not operating effectively. For example, when an agent is not able to satisfy commitments, or resources become overloaded or underutilized, the diagnosis component will be notified. The diagnosis component will then reconstruct the situation to understand why the problem has occurred and whether any assumptions used in the design of the organizational structure are no longer valid. If this is the case, the organizational design component responsible for this area of the organization will be notified so as to adapt the organization to the new situation. The organization component is also notified when resource failures occur and new resources enter the system. The learning component is connected to the monitoring and detection components to generate more precise descriptions of task performance characteristics, recognize coordination relations that are not apparent on the surface (such as seemingly unrelated tasks repeatedly co-occurring in the system within a fixed time period), and recognize problem phenomena that cannot be isolated by the diagnosis component based on a single problem instance.

An important part of the architecture not adequately reflected in this diagram is the ability of the agent designer through the problem-solving/task assessor component to introduce new and possibly domain-dependent coordination strategies via a programming language interface as described in such systems as [2,36,60,62], thus expanding the range of possible small group coordination strategies that can be implemented in this architecture. Our proposed approach will be for the task assessor to translate aspects of the representation of the user-defined coordination strategy into organizational knowledge that will appropriately condition the coordination component (e.g., what coordination relationships/mechanisms to use for particular tasks, what *a priori* commitments need to be initialized with their relative importance, etc.) and will signal the task assessor when certain conditions occur in the coordination process (e.g., not being able to meet a commitment, resource overload, etc.). The task assessor, as the result of being notified of the occurrence of an event either by the problem solver or by the coordination mechanism, will provide new organizational knowledge directives to the coordination component in order to implement the semantics of the user-defined coordination strategy. I feel that this goal of extensibility can be achieved given the right base-level coordination mechanisms, the appropriate abilities to tailor these mechanisms through organizational knowledge, and the capability to monitor and react to events occurring during the execution of the coordination mechanism.

As detailed in the introduction, whether agents are self-interested or cooperative has purposely not been indicated in these discussions. It is expected that the small-team agent coordination mechanisms, contained in this architecture, will support both self-interested and cooperative strategies, and also strategies for self-interested and cooperative agents to work together. To reiterate, I feel that even though the coordination mechanisms that have been developed to date for each type of agent are very different [28, 32, 53, 55, 68], there is more in common among cooperative and self-interested coordination mechanisms than currently believed, and this will become increasingly clear as self-interested agents are applied to combinatorially explosive problems [28, 31, 54] and function in large ensembles of agents.

This proposed multi-agent architecture should not be seen as a finished work but rather as an instance of how the ideas and functionality discussed in the first two sections of this paper can be computationally implemented.

4. Research Questions

Many of the components needed to fulfill this vision of robust and distributed information systems have already been designed, built, and evaluated. However, these evaluations have been limited to isolated components, a few example tasks, and small agent groups. Thus, fundamental questions remain to be answered: Is this a sufficient set of technologies with which to produce large and complex multi-agent systems? Can large-scale agent organizations be constructed by appropriately tailoring the dynamic coordination patterns of small groups of agents? Is it possible to have both a rich and complex coordination mechanism and infrastructure while simultaneously making this infrastructure open to being programmed? Can domain-specific coordination strategies be created that build upon the infrastructure and were not envisioned in the original design? What are the appropriate measures of system robustness and scalability?

In addition, there is a myriad of more detailed research questions that need to be answered: Is the TÆMS representational framework sufficiently powerful to allow reasoning about a wide range of computational activities? What is the appropriate interplay between local agent scheduling, dynamic small group agent coordination and organizational design and adaptation? How much situation-specificity can be achieved within a framework that is attempting to be very broad in the systems that it can represent? Are there issues of scale that will cause us to realize that a more complex view of control layers and their interaction needs to be developed? Can organizations be constructed where there is effective interaction among cooperative and self-interested agents? For what portion of an agent organization and for what situations is it worthwhile to design off-line, based on a description of the environment and likely tasks, versus on-line learning of the organization as a result of a series of local adaptations based on experience? Can effective diagnosis of ineffective or harmful behavior be based on a domain-independent representation of agent activities? How many instances will be needed for effective learning and how can knowledge learned in one part of the organization be transferred/related to other parts of the organization?

Obviously, this list of research questions is only a small subset of the issues that need to be addressed in order to be able to construct large, complex and robust distributed information systems.

5. Conclusions

The paper lays out many of the problems associated with the design of an agent architecture which has to operate in an open and large-scale multi-agent environment. In addition to laying out the problems, a method of approach for addressing these problems and a generic architecture based on this approach is put forward and a number of open research issues are presented. I believe that this architecture contains many of the key features necessary to support the next generation of sophisticated multi-agent applications. It contains five components — *local agent scheduling*, *multi-agent coordination*, *organizational design*, *detection and diagnosis* and *on-line learning* — that are designed to interact so that a range of different situation-specific coordination strategies can be implemented and adapted as the situation evolves. This proposed multi-agent architecture should not be seen as a finished work but rather as an instance of how a wide range of ideas developed by the Multi-Agent/DAI community can be integrated into a viable computational framework. There is no intent to say that this is the "right" way to do things but rather to spark a dialogue among

researchers about the software infrastructure needed to make the construction of large, complex and robust agent societies easier.

6. Acknowledgments

The reflections expressed in this paper about the type of agent architecture necessary to build the next generation of multi-agent applications are based on a long lineage of ideas that have come from both the Multi-Agent/DAI Laboratory at the University of Massachusetts, Amherst, and from the wider community. Special mention needs to go to Keith Decker, whose thesis work on GPGP is the underpinning of the proposed architecture. I also want to give credit to the following people that I have closely collaborated with, whose work directly relates to this paper: Norman Carver, Daniel Corkill, Edmund Durfee, Satoru Fujita, Alan Garvey, Eva Hudlicka, David Jensen, Susan Lander, Daniel Neiman, M. Nagendra Prasad, Tuomas Sandholm, Toshi Sugawara and Thomas Wagner.

References

1. R. Axelrod, *The Evolution of Cooperation*, Basic Books, 1984.
2. M. Barbuceanu and M. S. Fox, "COOL: A Language for Describing Coordination in Multi-Agent Systems," in *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press: Menlo Park, CA, pp. 17-24, June 1995.
3. N. Carver and V. Lesser, "A New Framework for Sensor Interpretation: Planning to Resolve Sources of Uncertainty," in *Proceedings of the Ninth National Conference on Artificial Intelligence*, pp. 724-731, August 1991.
4. N. Carver and V. Lesser, "The DRESUN Testbed for Research in FA/C Distributed Situation Assessment: Extensions to the Model of External Evidence," in *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press: Menlo Park, CA, pp. 33–40, June 1995.
5. C. Castelfranchi, "Commitments: From Individual Intentions to Groups and Organizations," in *AI and Theories of Groups & Organizations: Conceptual and Empirical Research*, M. Prietula (ed.),Working Notes of AAAI Workshop, 1993.
6. D. Cockburn and N.R. Jennings, "ARCHON: A Distributed Artificial Intelligence System for Industrial Applications," in *Foundations of Distributed Artificial Intelligence*, G.M.P. O'Hare and N.R. Jennings (eds.), John Wiley & Sons, Ch. 12, pp. 319-344, 1996.
7. S.E. Conry, K. Kuwabara, V. R. Lesser, and R. A. Meyer, "Multistage Negotiation for Distributed Constraint Satisfaction," *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6), November 1991.
8. D. D. Corkill and V. R. Lesser, "The Use of Meta-Level Control for Coordination in a Distributed Problem Solving Network," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, pp. 748-755, August 1983.
9. K. S. Decker, A. Pannu, K. Sycara, and M. Williamson, "Designing Behaviors for Information Agents," in *Proceedings of the First International Conference on Autonomous Agents*, February 1997.

10. K. S. Decker, K. Sycara, and M. Williamson, "Middle-Agents for the Internet," in *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, Japan, August 1997.

11. K. S. Decker and K. Sycara, "Intelligent Adaptive Information Agents," *Journal of Intelligent Information Systems*, 1997. (To appear.)

12. K Decker, "TÆMS: A Framework for Analysis and Design of Coordination Mechanisms," in *Foundations of Distributed Artificial Intelligence*, G. O'Hare and N. Jennings (eds.), Wiley Inter-Science, Ch. 16, 1995.

13. K. S. Decker and V. R. Lesser, "An Approach to Analyzing the Need for Meta-Level Communication," in *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Chambery, France, pp. 360-366, August 1993.

14. K. S. Decker and V. R. Lesser, "A One-Shot Dynamic Coordination Algorithm for Distributed Sensor Networks," in *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, pp. 210-216, July 1993.

15. K. S. Decker and V. R. Lesser, "Quantitative Modeling of Complex Environments," *International Journal of Intelligent Systems in Accounting, Finance, and Management*, Special issue on Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior, 2(4): 215-234, December 1993.

16. K. S. Decker and V. R. Lesser, "Designing a Family of Coordination Algorithms," in *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press: Menlo Park, CA, pp. 73-80, June 1995. (Longer version available as UMass CS-TR 94-14.)

17. K. S. Decker, *Environment Centered Analysis and Design of Coordination Mechanisms,* Ph.D. Dissertation, Computer Science Department, University of Massachusetts at Amherst, 1995.

18. K. S. Decker, "Task Environment Centered Simulation," in *Simulating Organizations: Computational Models of Institutions and Groups*, M. Prietula, K. Carley, and L. Gasser (eds.), AAAI Press/MIT Press, 1997.

19. K. S. Decker and Jinjiang Li, "Coordinated Hospital Patient Scheduling," University of Delaware, Department of Computer Science Technical Report 98-12, 1998.

20. E. H. Durfee, "Blissful Ignorance: Knowing Just Enough to Coordinate Well," in *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press: Menlo Park, CA, pp. 406-413, June 1995.

21. E. H. Durfee, V. R. Lesser, and D. D. Corkill, "Coherent Cooperation Among Communicating Problem Solvers," *IEEE Transactions on Computers*, 36(11): 1275-1291, November 1987.

22. E. H. Durfee and V. R. Lesser, "Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation," *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5): 1167-1183, September 1991.

23. K. Fischer, J.P. Muller and M. Pischel, "Unifying Control in a Layered Architecture," in *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press: Menlo Park, CA, pp. 446, June 1995.

24. T. Finin, R. Fritzson, D. McKay and R. McEntire, "KQML: An Information and Knowledge Exchange Protocol," *Knowledge Building and Knowledge Sharing*, K. Fuchi and T. Yokoi (eds.), Ohmsha and IOS Press, 1994.

25. A. Garvey, K. Decker, and V. Lesser, "A Negotiation-based Interface Between a Real-time Scheduler and a Decision-Maker," in *Proceedings of Workshop on Models of Conflict Management in Cooperative Problem Solving*, AAAI, Seattle, WA, July 1994. (Also appears as: University of Massachusetts, Computer Science Department Technical Report 94–08, January 1994.)

26. L. Gasser, "DAI Approaches to Coordination," *Distributed Artificial Intelligence: Theory and Praxis*, N.M. Avouris and L. Gasser (eds.), Kluwer Academic Publishers: Boston, pp. 31-51, 1992.

27. L. Gasser, Private communication, September 1997.

28. P.J. Gmytrasiewicz and E. H. Durfee, "A Rigorous, Operational Formalization of Recursive Modeling," in *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press: Menlo Park, CA, pp. 125–132, June 1995.

29. B.J. Grosz and C. L. Sidner, "Plans for Discourse," in *Intentions in Communication*, P.R. Cohen, J. Morgan, and M.E. Pollack (eds.), MIT Press: Cambridge, MA, pp. 417-444, 1990.

30. A. Haddadi, "Towards a Pragmatic Theory of Interactions," in *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press: Menlo Park, CA, pp. 133-139, June 1995.

31. J.Y. Halpern and Y. Moses, "Knowledge and Common Knowledge in a Distributed Environment," in *Proceedings of the Third ACM Conference on Principles of Distributed Computing*, 1984.

32. B.A. Huberman and T. Hogg, "The Behavior of Computational Ecologies," in *The Ecology of Computation*, B.A. Huberman (ed.), Amsterdam: North-Holland Publ., pp. 77-115, 1988.

33. N.R. Jennings, "Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems," *The Knowledge Engineering Review*, vol. 8, no. 3, pp. 223-250, 1993.

34. N.R. Jennings, "Coordination Techniques for Distributed Artificial Intelligence," in *Foundations of Distributed Artificial Intelligence*, G.M.P. O'Hare and N.R. Jennings (eds.), John Wiley & Sons, Ch. 6, pp. 187-210, 1996.

35. N. Jennings, "Controlling Cooperative Problem Solving in Industrial Multi-Agent Systems Using Joint Intentions," *Artificial Intelligence*, Vol. 75, No. 2, 1995.

36. K. Kuwabara, T. Ishida and N. Osato, "AgentTalk: Coordination Protocol Description for Multiagent Systems," in *Proceedings of the First International Conference on Multi-Agent Systems*, AAAI Press: Menlo Park, CA, pp. 455, June 1995.

37. B. Lâasri, H. Lâasri, S. Lander, and V. Lesser, "A Generic Model for Intelligent Negotiating Agents," *International Journal on Intelligent Cooperative Information Systems*, 1(2): 291-317, 1992.

38. S. Lander and V.R. Lesser, "Negotiated Search: Organizing Cooperative Search Among Heterogeneous Expert Agents," in *Proceedings of the Fifth International Symposium on Artificial Intelligence Applications in Manufacturing and Robotics*, December 1992.

39. V. Lesser, K. Decker, N. Carver, A. Garvey, D. Neiman, M. Nagendra Prasad and T. Wagner, "Evolution of the GPGP Domain-Independent Coordination Framework," University of Massachusetts at Amherst, Computer Science Department Technical Report 98-05, January 1998.

40. V. Lesser, B. Horling, F. Klassner, A. Raja, T. Wagner and S. XQ. Zhang, "BIG: A Resource-Bounded Information Gathering Agent," University of Massachusetts at Amherst, Computer Science Department, Technical Report 98-03, January 1998.

41. V. Lesser and L.D. Erman, "Distributed Interpretation: A Model and an Experiment," *IEEE Transactions on Computers — Special Issue on Distributed Processing*, Vol. C–29, 12, pp. 1144–1163, December 1980.

42. V. Lesser and D.D. Corkill, "Functionally-Accurate Cooperative Distributed Systems," *IEEE Transactions on Systems, Man, and Cybernetics — Special Issue on Distributed Problem-Solving*, Vol. SMC–11, No. 1, pp. 81–96, Jan. 1981.

43. V. Lesser, "A Retrospective View of FA/C Distributed Problem Solving," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 6, pp. 1347-1362, Nov./Dec. 1991.

44. V. Lesser and D. D. Corkill, "The Distributed Vehicle Monitoring Testbed," *AI Magazine*, 4(3): 63-109, Fall 1983.

45. T. Moehlman, V. Lesser, and B. Buteau, "Decentralized Negotiation: An Approach to the Distributed Planning Problem," *Group Decision and Negotiation*, 1(2): 161-192, 1992.

46. M. Nagendra Prasad and V. Lesser, "The Use of Meta-level Information in Learning Situation-Specific Coordination," in *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, August 1997.

47. M.V. Nagendra Prasad, K. Decker, A. Garvey and V. Lesser, "Exploring Organizational Designs with TAEMS: A Case Study of Distributed Data Processing," in *Proceedings of the Second International Conference on Multi-Agent Systems,* AAAI Press: Menlo Park, CA, pp. 283–290, 1996.

48. D. Neiman, D.W. Hildum, V.R. Lesser, and T.W. Sandholm, "Exploiting Meta-Level Information in a Distributed Scheduling System," in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA, August 1994.

49. T. Oates, M. V. Nagendra Prasad and V. Lesser, "Cooperative Information Gathering: A Distributed Problem Solving Approach," in *IEE Proceedings on Software Engineering*, Special Issue on Agent-based Systems, vol. 144, no. 1, 1997.

50. L. Obrst, M. Woytowitz, D. Rock, S. Lander, K. Gallagher and K. Decker, "Agent-Based Integrated Project Teams," in *Proceedings of the EIM97 Engineering Information Management Symposium* of the ASME Design Engineering Technical Conferences, Sacramento, CA, September 1997.

51. K. Ramamritham and J. A. Stankovic, "Scheduling Algorithms and Operating Systems Support for Real-Time Systems," in *Proceedings of the IEEE*, pp. 55-67, Jan. 1994.

52. A.S. Rao and M.P. Georgeff, "BDI agents: From Theory to Practice," in *Proceedings of the First International Conference on Multi-Agent Systems*, pp. 312–319, AAAI Press: Menlo Park, CA, June 1995.

53. J. Rosenschein and G. Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*, Cambridge, MIT Press: MA, 1994.

54. T. Sandholm and V. Lesser, "Coalitions Among Computationally Bounded Agents," *Artificial Intelligence*, Special Issue on Principles of Multiagent Systems, 1997.

55. T. Sandholm and V. Lesser, "Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework," in *Proceedings of the First*

International Conference on Multi-Agent Systems, AAAI Press: Menlo Park, CA, 1995.

56. D. Steiner, "IMAGINE: An Integrated Environment for Constructing Distributed Artificial Intelligence Systems," in *Foundations of Distributed Artificial Intelligence*, G.M.P. O'Hare and N.R. Jennings (eds.), John Wiley & Sons, Ch. 13, pp. 344-366, 1996.
57. T. Sugawara and V. Lesser, "Learning to Improve Coordinated Actions in Cooperative Distributed Problem-Solving Environments," *Machine Learning*, Kluwer Academic Publishers. (To appear, 1998.)
58. T. Sugawara and V. Lesser, "Learning Control Rules for Coordination," in *Multi Agent and Cooperative Computation '93*, pp. 121-136, 1993.
59. T. Sugawara and V. Lesser, "On-Line Learning of Coordination Plans," in *Twelfth Annual Workshop on Distributed Artificial Intelligence*, 1993.
60. S. M. Sutton, Jr. and L. J. Osterweil, "The Design of a Next-Generation Process Language," in *Proceedings of the Joint 6th European Software Engineering Conference and the 5th ACM SIGSOFT Symposium on the Foundations of Software Engineering*, Springer-Verlag: Zurich, Switzerland, September 1997.
61. K. Sycara, K. Decker, A M. Williamson and D. Zeng, "Distributed Intelligent Agents," *IEEE Expert*, vol 11, number 6, 1996.
62. M. Tambe, "Agent Architectures for Flexible, Practical Teamwork," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Providence, July 1997.
63. A. S. Tannenbaum, *Distributed Operating Systems*, Prentice-Hall, 1995.
64. R. Turner, "The Tragedy of the Commons and Distributed AI Systems," University of New Hampshire, Computer Science Department Technical Report 93-01. (http://cdps.umcs.maine.edu/Papers/1993/TofCommons/TR.html), 1993.
65. F. von Martial, "Coordinating Plans of Autonomous Agents," *Lecture Notes in Artificial Intelligence*, no. 610, Springer-Verlag: Berlin, 1992.
66. T. Wagner, A. Garvey, and V. Lesser, "Complex Goal Criteria and Its Application in Design-to-Criteria Scheduling," in *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997.
67. T. Wagner, A. Garvey, and V. Lesser, "Criteria Directed Task Scheduling," in *International Journal of Approximate Reasoning,* Elsevier Science Inc., Vol. 19, pp. 91–118.
68. M. Wellman, "A Market-Oriented Programming Environment and Its Application to Distributed Multicommodity Flow Problems," *Journal of Artificial Intelligence Research*, vol. 1, pp. 1-22, 1993.

Footnotes

[1]As part of this framework, I see agents eventually using high-level content languages for rich and succinct communication with other agents. I also foresee that the next generation of distributed operating systems and network communication support will be more reflective and able to adapt to changing resource availability [51]. This more sophisticated operating environment opens up the possibility of dialogues between the multi-agent system (concerned with performance and adaptability from the application level perspective) and the operating system (concerned with performance and adaptability at the resource level); the goal of this interaction would be to achieve the most appropriate configuration of resources and computational processing. The implications of these capabilities on an agent's architecture will not be considered in further discussion.

[2]There is a wide range of these subproblem interdependencies—some of which have been categorized such as *facilitates* relationship [15] where the solution to one subproblem, if it is available at the point another subproblem is executed, will speed up the execution of that subproblem or increase the likelihood that a higher quality result will be produced. Another example is the *favor* relationship [65] where the solution to two subproblems requires the use of the same resource, and by jointly solving the combined subproblems the overall usage of the resource by the two agents can be reduced significantly. These two can be characterized as soft relationships because they only affect performance issues. Other relationships, such as *enables* which denote when the result from one problem-solving activity is required to perform another, can be considered hard relationships because they indicate relationships that must be enforced if problem-solving is to be correct.

[3]There are numerous examples of social phenomena such as the *tragedy of the commons* [1, 64] where agents basing their decisions from a totally self-interested perspective eventually leads to disastrous consequences for every agent in the society.

[4]An early example of this perspective was the coordination mechanism proposed in [8] which involved skeptical agents. The idea was to seed the agent society with a small number of agents who would require very strong evidence to accept the views/directives of other agents. In this way, it was reasoned that the society as a whole would always possess some ability to recover from incorrect decisions that at the time seemed reasonable to most of the agents.

[5]From the limited experience I have had in scaling up coordination to larger groups of agents [22], it seems that small agent groups will probably involve less than ten agents. The size of the group is dictated by the availability of computational and communication resources which will allow the detailed scheduling of local agent activities so that they are aligned with the activities of other agents in the group.

[6]The GRATE distributed planning protocol [35], which combines commitment importance and temporal ordering constraints with symbolic reasoning about joint commitment, has a lot of similarity with GPGP [16]. GPGP has more sophisticated quantitative reasoning whereas GRATE has more sophisticated symbolic reasoning.

[7]When these relationships exist across tasks in other agents they are called NLEs (non-local relationships). NLEs can either be specified by an agent *a priori* by indicating as part of the task structure that a task of a particular type (name) in another agent will have a specified relationship with one of its tasks, or can be discovered as a result of agents exchanging information about their task structures. In this case, the agents need to provide

code that will take domain-specific attributes that can be optionally attached to the task name to determine what task relationships exist among the tasks of both agents.

[8]Through the use of task-resource relationships, we can introduce information necessary to understand how the concurrent use of a resource by multiple agents will affect agent performance. In its simplest use, it permits the implementation of a coordination mechanism that guarantees that non-sharable resources will be used by only one agent at a time [19].

[9]Coordination in GPGP is achieved through the use of *commitments*, that is, inter-agent contracts to perform certain tasks by certain times. These commitments are normally constructed dynamically as a result of a dialogue among the coordination modules of agents. By allowing these commitments to be specified *a priori* as part of an agent's TÆMS task structure representation of its activities, low overhead coordination can be achieved among agents. For example, *a priori* commitments could indicate that an agent can expect another agent to generate a specific result and transmit it to this agent by a certain time [47]. The transmitting agent, in turn, has an *a priori* commitment to generate the results by a specific time. In this way, agent activities can be coordinated without the agents exchanging information about their current activities, and then negotiating over a suitable commitment. Additionally, certain soft task relationships that require the cooperation of other agents may not be worthwhile to achieve in the current environment, because of the coordination overhead necessary to establish them. By indicating that these relationships should not be achieved, the coordination strategy can be made situation-specific.

[10]The task assessor module can be implemented without significant computational overhead if the underlying problem solver's control regime is not data-dependent or as in [50] where a simplified version of TÆMS is already integrated into the control of domain problem solving. The more opportunistic the problem solving the greater the effort required of the task assessor to generate a task structure that is representative of the possible activity patterns of the problem-solving component, is indicative of the reasonable variations in performance that can be scheduled for, and specifies the points in the activity pattern where interaction with other agents are appropriate. An early and simplified example of such a task assessor was used in the predecessor work to GPGP called PGP [22]. Moderately complex task assessor modules were recently constructed for use in real-time control of a single-agent system in [40] and for multi-agent control in [4].