Task Environment Centered Simulation *

Keith S. Decker
Department of Computer Science
University of Massachusetts
DECKER@CS.UMASS.EDU

April 2, 1995

The design of organizations or other coordination mechanisms for groups of agents depends in many ways on the agent's task environment. Two of these dependencies are on the structure of the tasks and on the uncertainty in the task structures. The task structure includes the scope of the problems facing the agents, the complexity of the choices facing the agents, and the the particular kinds and patterns of interrelationships that occur between tasks. A few examples of environmental uncertainty include uncertainty in the *a priori* structure of any particular problem-solving episode, in the actions of other agents, and in the outcomes of an agent's own actions. These dependencies hold regardless of whether the system comprises just people, just computational agents, or a mixture of the two. For example, the presence of both uncertainty and high variance in a task structure can lead a system of agents to perform better by using coordination algorithms that adapt dynamically to each problem-solving episode [8, 9]. Designing coordination mechanisms also depends on non-task characteristics of the environment such as communication costs, and properties of the agents themselves. Representing and reasoning about the task environment must be part of any computational theory of coordination.

TÆMS (Task Analysis, Environment Modeling, and Simulation) was developed as a framework with which to model and simulate complex, computationally intensive task environments at multiple levels of abstraction and from multiple viewpoints. It is a tool for building and testing computational theories of coordination. TÆMS is compatible with both formal computational agent-centered approaches and experimental approaches. The framework allows us to both mathematically analyze (when possible) and quantitatively simulate the behavior of multi-agent systems with respect to interesting characteristics of the computational task environments of which they are part. We believe that it provides the correct level of abstraction for meaningfully evaluating centralized, parallel, and distributed control algorithms, negotiation strategies, and organizational designs.

This chapter will briefly describe the TÆMS modeling framework for representing abstract task environments, concentrating particularly on its support for simulation. I will describe how to model each of several different multi-agent problem-solving environments, such as

*This work was supported by DARPA contract N00014-92-J-1698, Office of Naval Research contract N00014-92-J-1450, and NSF contract IRI-9321324. The content of the information does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

distributed situation assessment, hospital patient scheduling, multi-physician consultation, airport ground services management, and multi-agent information retrieval. I will also discuss the generation of episodes within environments for simulation.

1 Background

Artificial Intelligence, growing as it has from the goal of modeling *individual* intelligence, or at least replicating or augmenting it, has focused primarily on representations of individual choice and action. A large effort has gone into describing the principled construction of agents that act rationally and predictably based on their beliefs, desires, intentions, and goals [5, 36]. Fairly recently, researchers concerned with real-world performance have also realized that Simon's criticisms and suggestions about economics [37, 27, 38] also hold for many realistically situated individual agents—perfect rationality is not possible with bounded computation [21, 1, 33, 15]. Distributed AI has too often kept the individualistic character of its roots, and focused on the principled construction of individual agents. It hasn't even, so far, really concerned itself with the questions of bounded rationality in real-time problem solving when it comes to the principled construction of individual agents. Worst of all, it has failed yet to bring the environment to center stage in building and analyzing distributed problem solving systems.

In contrast, the organizational science community has since the 60's (e.g. [22]) regarded the task environment as a crucial, central variable in explaining complex systems, and a whole branch of research has grown up around it (contingency theory). Representations in this community are rarely mathematically formal in nature but rather try to present very rich descriptions using terms such as uncertainty, decomposability, stability, etc.

TÆMS, as a framework to represent coordination problems in a formal, domain-independent way, is unlike any existing computational representation that is focussed on coordination issues. The form of the framework is more detailed in structure than many organizational-theoretic models of organizational environments, such as Thompson's notions of pooled, sequential, and reciprocal processes [42], Burton and Obel's linear programs [2], or Malone's queueing models [26], but is influenced by them, and by the importance of environmental uncertainty and dependency that appear in contingency-theoretic and open systems views of organizations [22, 14, 40, 34]. As a problem representation for computational tasks, it is richer and more expressive than game theory [32, 45, 18] or team theory [20] representations. For example, a typical game or team theory problem statement is concerned with a single decision; a typical TÆMS objective problem solving episode represents the possible outcomes of many sequences of choices that are interrelated with one another (e.g., "schedules"). TÆMS can represent a game theoretic problem, and we could boil down a single decision made by an agent faced with a TÆMS task structure into a game theoretic problem.[1] Because TÆMS is more expressive, we can use it to operationalize some of the rich but informal concepts of organizational science (such as *decomposability* in Section 6). Another difference between TÆMS and traditional distributed computing task representations is that TÆMS indicates that not all tasks in an episode need to be done.

As a tool for building and testing computational theories of coordination, the TÆMS

[1] TÆMS does not say how agents make their decisions. It is perfectly reasonable for an agent to use game-theoretic reasoning processes.

framework can, for example, support the construction of ACTS theory instances [3]. In ACTS theory organizations are viewed as collections of intelligent agents who are cognitively restricted, task oriented, and socially situated. TÆMS provides ways to think about and represent environmental constraints (task characteristics and social characteristics involving communication links and what information and what possible actions are available to what agents). While simple models can some times be solved analytically [8, 9], many complex models require simulation techniques. Compared to other organizational simulations such as [25, 23], TÆMS provides a much more detailed model of task structures, and does not provide a fixed agent model.

1.1 Features of TÆMS

The unique features of TÆMS include:

- The explicit, quantitative representation of task interrelationships. Both hard and soft, positive and negative relationships can be represented. When relationships in the environment extend between tasks being worked on by separate agents, we call them *coordination relationships*. Coordination relationships are crucial to the design and analysis of coordination mechanisms. The set of relationships is extensible.

  An example of a 'hard' relationship is enables. If some task A enables another task B, then A must be completed before B can begin. An example of a 'soft' relationship is facilitates. If some task A facilitates a task B, then completing A before beginning work on B might cause B to take less time, or cause B to produce a higher-quality result, or both. If A is *not* completed before work on B is started, then B can still be completed, but might take longer or produce a lower quality answer than in the previous case. In other words, completing task A is not necessary for completing task B, but it is helpful.[2]

- The representation of the structure of a problem at multiple levels of abstraction. The highest level of abstraction is called a *task group*, and contains all tasks that have explicit computational interrelationships. A *task* is simply a set of lower-level subtasks and/or executable methods. The components of a task have an explicitly defined effect on the quality of the encompassing task. The lowest level of abstraction is called an executable *method*. An executable method represents a schedulable entity, such as an instance of a human activity at some useful level of detail, for example, "take an X-ray of patient 1's left foot". For a computational agent, a method could also be a blackboard knowledge source instance, a chunk of code and its input data, or a totally-ordered plan that has been recalled from memory and instantiated for a task.

- TÆMS makes very few assumptions about what an 'agent' is like cognitively. TÆMS defines an agent as a locus of subjective beliefs (or state) and actions (executing methods, communicating, and acquiring subjective information about the current problem solving episode). This is important because the study of principled agent construction is a very active area. By separating the notion of agency from the model of task environments, one

[2] For example, imagine that that your task is to find a new book in a library, and you can do this either before or after the new books are unpacked, sorted, and correctly shelved.

does not have to subscribe to a particular agent architecture (which one would assume will be adapted to the task environment at hand). This separation also allows a user to ask questions about the inherent social nature of the task environment at hand (allowing that the concept of society may arise before the concept of individual agents [17]). Such a conception is unique among computational approaches.

- The representation of the task structure from three different viewpoints. The first view is a *generative* model of the problem solving episodes in an environment—a statistical view of the task structures. The second view is an *objective* view of the actual, real, instantiated task structures that are present in an episode. The third view is the *subjective* view that the agents have of objective reality. The subjective view is derived from the social situation, i.e., the organizational role of an agent may partly determine what part of the whole task structure that agent sees (and with what certainty).

  For example, in this chapter we will discuss a hospital scheduling environment. The generative model describes how to generate a hospital scheduling episode (perhaps one day's worth of work). The generative model will describe how often new patients arrive at the hospital, and what procedures need to be done on them (for instance, Patient 12 needs a foot X-ray and physical therapy). The resulting objective task structure episode describes, from Nature's global omniscient point-of-view, what tasks are possible as time goes forward, and how those tasks are related (perhaps the physical therapy must come after the X-ray). Finally, the generative model also generates a subjective model that describes what portions of the task structures become available to the agents and at what time. For example, only the nurse in charge of the patient's nursing unit will initially have access to the tests prescribed by the doctor to that patient; the nurse must explicitly inform ancillary hospital units such as the X-ray unit about the need for X-ray tasks using the patient.

  The TÆMS representation of an objective task structure is *not* intended as a schedule or plan representation, although it intentionally provides and relies on much of the information that would go into such uses.

- TÆMS allows us to clearly specify concepts and subproblems important to multi-agent and AI scheduling approaches. For example, we can clearly represent the difference between "anytime" [1] and "design-to-time" [15] algorithms in TÆMS.

- This chapter will focus mostly on *computational* task environments, where agents do not require shared physical resources. However, I will also describe extensions of TÆMS to represent shared physical resource constraints.

As I have mentioned, TÆMS provides a system for simulating environments: generating random episodes, providing subjective (socially situated) information to the agents, and tracking their performance. The TÆMS simulator is written in portable CLOS (the Common Lisp Object System) and uses CLIP [44] for data collection. Simulation is a useful tool for learning parameters to control algorithms, for quickly exploring the behavior space of a new control algorithm, and for conducting controlled, repeatable experiments when direct mathematical analysis is unwarranted or too complex. The simulation system we have built for the direct

execution of models in the TÆMS framework supports, for example, the collection of paired response data, where different or ablated coordination or local scheduling algorithms can be compared on identical instances of a wide variety of situations. For example, we have used simulation to explore the effect of exploiting the presence of *facilitation* between tasks in a multi-agent real-time environment where no quality can be achieved after a task's deadline [7]. The generative environmental characteristics there included the mean inter-arrival time for tasks, the likelihood of one task facilitating another, and the strength of the facilitation (see the next section).

We validated the TÆMS framework by building a detailed model of the complex DSN environment of the Distributed Vehicle Monitoring Testbed (DVMT). Simulations of simplified DSN models show many of the same characteristics as were seen in the DVMT [12]. We have also described models of many other environments: hospital patient scheduling, the prisoners' dilemma, airport resource management, multi-agent Internet information gathering, and the pilot's associate. Some of these will be summarized in this chapter. Finally, we have validated our framework by allowing others to use it in their work—on design-to-time scheduling, on parallel scheduling, on the diagnosis of errors in local area networks, on Internet information gathering, and on learning when to apply particular coordination mechanisms. We encourage its use in exploring computational theories of human or mixed human/computer organizations as well.

2 Building a TÆMS model

In this section I will give a brief example of the construction of a TÆMS objective level model for a single episode. Typically a TÆMS user will begin with a set of representative episodes (problem instances) to model at the objective level. At this point the user decides:

- at what level of detail he or she will model the tasks and methods (depends on the questions being asked)
- what the performance measures will be (TÆMS provides duration and quality, but quality may mean different things in different environments (or even be a vector)
- what task interrelationships are needed (TÆMS currently has definitions for enables, facilitates, bounded facilitates, simple facilitates, hinders, precedes, causes, shares-results, cancels, uses, limits, consumes, consumable, replenishes, full, inhibits, and requires-delay [6]).

After building several episodes, it is easier to move on to specifying a generative model and a subjective model. The generative model deals with the question of how might those episodes actually come about, and what parts of them are routine and what parts are extraordinary. Being able to generate a large number of reasonable episodes is important especially when using simulation so that one does not jump to conclusions about only a few hand-picked examples. A generative model needs to generate both objective and subjective information for each episode. The subjective information describes, for example, what parts of the task structure are seen by

agents with different organizational roles, and what information those agents can access about the task structure they perceive.[3]

Let's build a very small example using a Post Office task [43] (we'll give much larger examples later in the chapter). A TÆMS task structure is specified at various levels of abstraction. Figure 1 shows a task structure for going to the post office to buy stamps. A directed acyclic graph of tasks connected by subtask relationships (here, it is a tree) is called a *task group*. Each task group has an identified root task (here, it is T1). Often agents know about many task groups simultaneously. A common performance measure is for the agents to attempt to maximize the sum of qualities achieved for all known task groups. Of course, not all task groups need to have equal importance, not all agents will know about all task groups, not all agents need to have the same performance criteria, task groups can have deadlines, and new task groups might appear at any time—we'll give some more complex examples later, and keep this one simple. At the leaves of the task group are tasks called *executable methods*. These are the only possible actions represented directly in the task structure. Two other classes of actions that agents can take are *communication* actions and *information gathering* actions. Communications can be at both the domain level (of partial results of tasks) or at a meta level (communications about tasks, such as commitments to do things by a certain time). Information gathering trades off an agent's time for better or newer information about the objective problem solving structure.

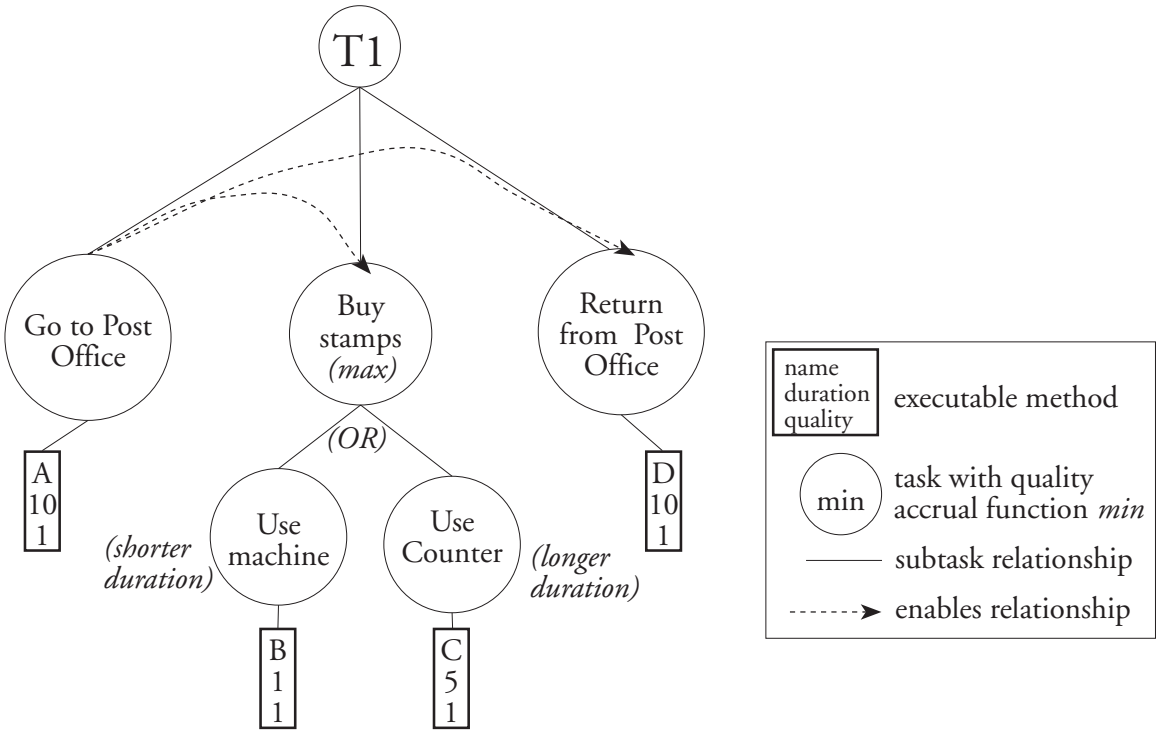

Figure 1: One possible instance of an objective task structure for buying stamps at the post office. Unless otherwise indicated, all quality accrual functions are *min* (OR).

Each executable method has an initial duration and an initial maximum quality. When there are no interrelationships, if an agent chooses the action of executing a method, it will take the

[3] An example of some of this sort of information are the task decomposition schemes defined by Lin [24].

amount of time as specified by the initial duration and produce the initial maximum "quality". Quality of a non-executable task is some function of the qualities of the subtasks (typically max (OR), min (AND), sum or mean). Task groups may have hard or soft deadlines—no quality can accrue for a task after a hard deadline. A good example of the difference between generative, objective, and subjective views is the specification of "duration". From the generative view, the duration of a task is often a statistical distribution[4]. From the objective view, the duration of some specific task instance is a single value. From the subjective view, the agent usually doesn't know that precise value, all that it knows is the original distribution from which the objective duration was drawn.

In Figure 1 the objective task group associated with buying stamps has the following structure: the root task T1 to buy stamps at the post office; subtasks to go to the post office, buy the stamps, and return from the post office; and two ways to buy the stamps at the post office that include using the stamp machine in the lobby or waiting in line to buy them at the post office counter. Each of these lowest-level tasks have executable methods associated with them, and each method has an initial duration and maximum quality. On top of this basic task/subtask structure, there can be task interrelationships ("non-local effects", or NLE's) such as enables. The enables NLE from "go to post office" to "buy stamps" is a shorthand that indicates that the enabling task enables all the executable methods below the enabled task (in Figure 1, these are methods B and C). Every NLE has a precise quantitative definition which describes changes in duration and/or quality of the effected method. For example, the definition of the enables effect on maximum quality looks like this:

$$\mathsf{enables}_Q(T_a, M, t, d, q, \theta) = \begin{cases} 0 & t < \Theta(T_a, \theta) \\ q & t \geq \Theta(T_a, \theta) \end{cases} \quad (1)$$

$\mathsf{enables}_Q$ indicates that this is the effect on maximum quality, T_a is the enabling task, M is the method being enabled, t is the current time, d the current duration of M, q the current maximum quality of M, and θ is a threshold parameter only for enables. The definition states that the current maximum quality of the enabled method M is 0 until such time as the quality of T_a rises above the threshold θ.[5] We have built a large collection of these relationship definitions that we often reuse, and we add to the collection whenever we find new relationships while modeling new environments.

Now let's look at an episode where there are two task groups: one (T1) to buy stamps, and another (T2) to mail a package at the post office (Figure 2). Once an agent waits in line to get to the counter, the agent may *both* buy stamps and mail a package, so these two tasks facilitate one another. The formal definition of facilitates [10, 6] takes two parameters, one for the change in duration, and one for the change in quality. In the postal example, there is no change in quality, but a large change in duration (once you wait in line to buy stamps at the counter, it is a short time to also mail a package, and visa versa). This particular episode is used by von Martial to illustrate a "favors" relationship, where the agent who needs to mail a package can reasonably offer to buy stamps for another agent. Note that it may *not* be reasonable for an agent who needs to buy stamps to offer to mail another agent's package, because the stamp-buying agent has the option of a much quicker method of buying stamps (using the machine) than waiting in line (which is required for mailing the package).

[4] Obviously different classes of tasks can have different duration distributions.

[5] The helper function $\Theta(T, \theta)$ returns the earliest time that the quality of T is greater than θ.

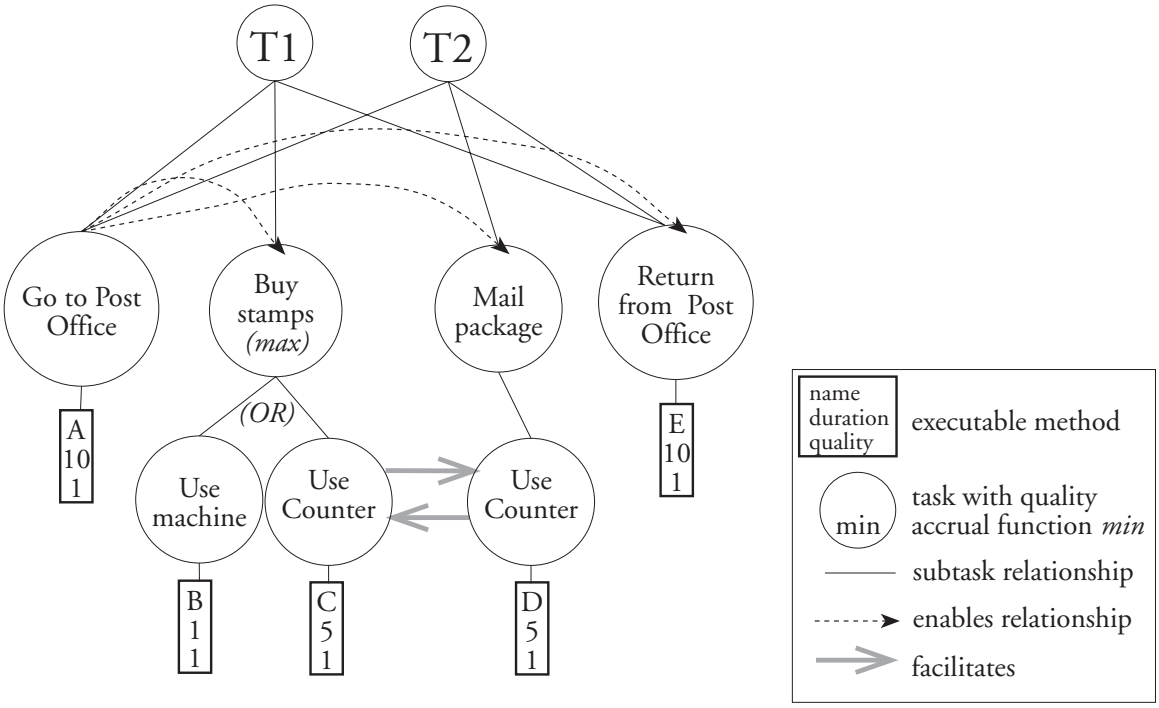

Figure 2: Possible objective task structures for buying stamps (T1) and mailing a package (T2) at the post office.

This has been a very simple example; we often work with much larger structures and many more agents. Note that we can represent both structures where the agents have a considerable number of choices in their actions (and need fairly complex agent models) as well as task structures where the agents are highly constrained in their actions [3]. We can also represent when an agent's capabilities or organizational role does not allow the agent to take certain actions by not creating an executable method for that agent (see the more complex examples later in this chapter).

3 Example TÆMS Models

The most detailed and analyzed TÆMS models are our simple and complex models of distributed sensor networks (DSN) and the Distributed Vehicle Monitoring Testbed (DVMT). Our models include features that represent approximate processing, faulty sensors and other noise sources, low quality solution errors, sensor configuration artifacts, and vehicle tracking phenomena such as training and ghost tracks. This work represented the first detailed analysis of a DSN, and the first quantitative, statistical analysis of any Distributed AI system outside Sen's work on distributed meeting scheduling for two agents [35]. This was important because much of the earlier work in the area had been ad hoc, anecdotal, or based on a small number of hand-constructed examples. The details of this work have been reported elsewhere [8, 9, 6]. We have been working on models of several other, quite different, task environments.

3.1 Hospital Scheduling

This description is from an actual case study [31]:

> *Patients in General Hospital reside in* units *that are organized by branches of medicine, such as orthopedics or neurosurgery. Each day, physicians request certain tests and/or therapy to be performed as a part of the diagnosis and treatment of a patient. [. . .] Tests are performed by separate, independent, and distally located* ancillary departments *in the hospital. The radiology department, for example, provides X-ray services and may receive requests from a number of different units in the hospital.*

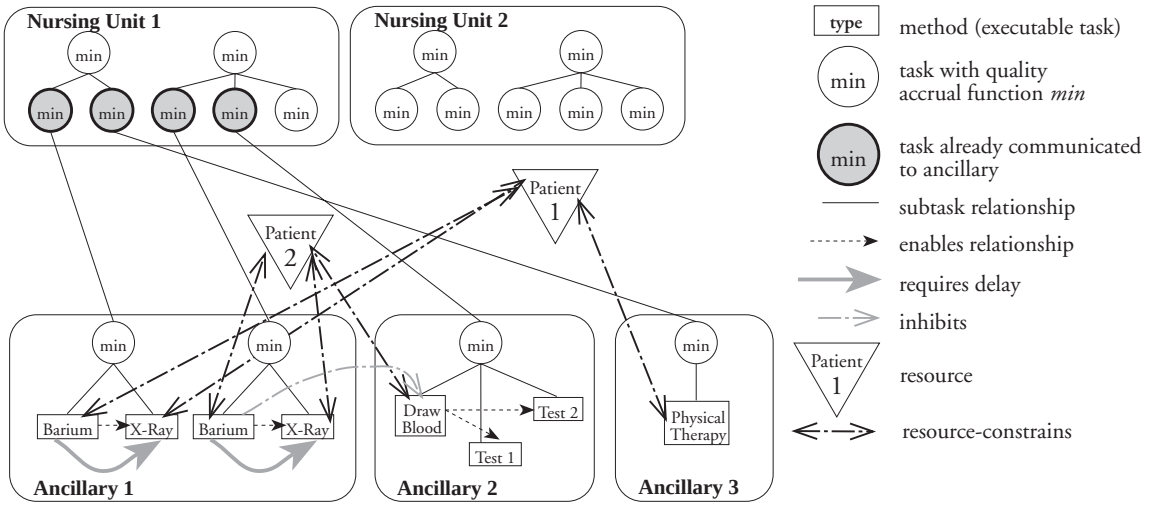

Figure 3: High-level, objective task structure and subjective views for a typical hospital patient scheduling episode. The top task in each ancillary is really the same objective entity as the unit task it is linked to in the diagram.

Furthermore, each test may interact with other tests in relationships such as enables, requires−delay (must be performed after), and inhibits (test A's performance invalidates test B's result if A is performed during specified time period relative to B). Note that the unit secretaries (as scheduling agents) try to minimize the patients' stays in the hospital, while the ancillary secretaries (as scheduling agents) try to maximize equipment use (throughput) and minimize setup times.

A generative level model of this environment would focus on the mean time between patient arrivals, and the number and distribution of patient tests. Other generative parameters would include the specification of the 'agents' in this environment: the number of nursing units and the number and type of ancillaries and their associated resources (i.e., how many X-ray machines can be used in parallel by ancillary 1). Thus an episode generator for Ow's hospital would have a fixed number of nursing units and ancillaries, and a fixed set of test templates corresponding to taking X-rays, physical therapy, blood tests, etc. These templates would also contain duration distributions on how long it takes to take the x-ray, etc. Then, we would postulate an arrival rate for patients, and for each patient a distribution of tests (instantiated

from the set of templates). Thus when each patient arrives, linked to that patient is a unique task structure of what needs to be done to that patient.

Figure 3 shows a hand-generated example objective TÆMS task structure corresponding to an episode in this domain, and the subjective views of the unit and ancillary scheduling agents after four tests have been ordered. I use *min* (AND) to represent quality accrual because in general neither the nursing units nor ancillaries can change the doctor's orders—all tests must be done as prescribed. This figure explicitly represents the patient as a non-sharable resource (resource − constrains represents the pair of NLE's uses and limits). Note that the patient is not needed for all portions of all tests (i.e., the blood work after the blood samples have been drawn). Also note that I have not represented the patient's travel times in any way—this is an important difference from the airport environment described in the next section. The reason for not representing this is that the scale of the problem is such that patient travel time can be fixed beforehand (auxiliaries can assume that the patients will be delivered from their rooms rather than from other auxiliaries). I will go so far as to point out the future implications of this downplay of travel times—this hospital will have to change its organization, i.e. its coordination and scheduling mechanisms, if it has to face significant travel delays with *more than one ancillary*. This could happen if, for example, it is a smaller hospital and cannot afford its own MR scanner or CAT scanner and doctors take to prescribing MR and CAT scans at another hospital. Patients requiring *both* MR and CAT scans now have a significant facilitates effect in their task structures to do both scans nearly consecutively (potentially affecting the coordination structures of *both* hospitals).

The requires − delay relationship says that a certain amount δ of time must pass after executing one method before the second is enabled. I'll repeat the definition I gave in the first chapter here:

$$\text{requires}-\text{delay}(T_a, M, t, d, q, \delta) = \begin{cases} [\mathbf{d}_0(M), 0] & \text{Start}(M) < \text{Finish}(T_a) + \delta \\ [\mathbf{d}_0(M), \mathbf{q}_0(M)] & \text{Start}(M) \geq \text{Finish}(T_a) + \delta \end{cases} \quad (2)$$

Examining the hospital's current coordination structure is enlightening because it shows a mismatch between the structure and the current hospital environment (this mismatch having triggered the study in the first place). From this mismatch we can guess at how the environment has changed over time, assuming that the current hospital structure was in fact a good structure when it was originally put into place.[6]

The current hospital structure is described by Ow as follows [31]:

> *After receiving a request from a physician, the unit secretary conveys the test request to the secretary of the relevant ancillary department. In turn, the ancillary secretary determined the appropriate time for the test to be run. The unit secretary is notified immediately prior to that scheduled time slot and not sooner. The actual time the patient is scheduled is known only to the ancillary secretary (i.e., it is not relayed back to the requesting unit). Since each ancillary schedules independently (and without knowledge) of all other ancillaries, conflicts arise when a patient is scheduled in overlapping (or nearly overlapping) time slots in different ancillaries. Such conflicts must be resolved by the unit secretaries. However, as the unit secretaries are made*

[6] Ow calls this the *sympathetic structure*[31].

aware of the scheduled ancillary times only when the request to "deliver" the patient comes from the ancillary, little slack time remains to resolve scheduling conflicts and delays, This can disrupt the care of the patient.[7]

While this structure seems sorely lacking when compared to the current environment, it may at one time have been a reasonable, low overhead arrangement. It may be that in the past doctors ordered fewer tests on less complex ancillary equipment (there has been quite an explosion in medical technology in the last decade) and (concomitantly) interfering relationships between these technologies. The structure is adapted for a different task environment.

Ow, et al. describe a new set of coordination mechanisms, using computer support, that are better adapted to the hospital's current task environment. Two types of computer agents are defined—a *unit subsystem* that collects, disseminates, and monitors the test requests and resulting schedules for the patients in a unit, and an *ancillary subsystem* that receives test requests from the unit subsystems and schedules them. Two communication protocols rest on this structure:

Primary: The unit subsystems inform the appropriate ancillary about the test request (transmitting part of the task structure). The request includes a soft deadline (the 'flow due date') for the task. The ancillary will schedule the test immediately, using a shared primary performance criteria of not missing the deadline, and secondary local criteria such as minimizing setups, maximizing throughput, etc. All other criteria being equal, the ancillary picks the earliest slot. As soon as the slot is chosen, the scheduled slot (what Generalized Partial Global Planning (GPGP) terms a *commitment*) is communicated back to the unit subsystem.

Secondary: As soon as a slot is scheduled for a patient at an ancillary, the slot (commitment) is *broadcast* to all other ancillaries, thus effectively blocking that slot from consideration by the other ancillaries for any tests on that patient (as indicated in the diagram by the resource−constrains relationships.

One thing to note about this new system is that it does not explicitly handle interactions between tests at different ancillaries. The GPGP family of coordination algorithms could be configured to respond to these interactions as well as eliminate broadcast communication.

3.2 Airport Resource Management

The UMASS ARM (Airport Resource Management [19]) and Dis-ARM (Distributed ARM [29]) systems solve airport ground service scheduling problems. The function of such systems is to ensure that each airport flight receives required ground servicing (gate assignment, baggage handling, catering, fuel, cleaning, etc.) in time to meet its arrival and departure deadlines [29]:

The supplying of a resource is usually a multi-step task consisting of setup, travel, and servicing actions. Each resource task is a subtask of the airplane servicing supertask.

[7] Additional problems may also ensue. For example, in some cases the *sequence* of multiple tests are important—a wrong sequence can result in patient stay delays as the residual effects of one test may influence the earliest start time of another test [31].

There is considerable parallelism in the task structure: many tasks can be done simultaneously. However, the choice of certain resource assignments can often constrain the start and end times of other tasks. For example, selection of a specific arrival gate for a plane may limit the choice of servicing vehicles due to transit time from their previous servicing locations and may limit refueling options due to the presence or lack of underground fuel tanks at that gate. For this reason, all resources of a specific type can not be considered interchangeable in the airport ground service scheduling domain.

The generative model for this environment includes the nominal flight schedule, a model of the errors in this schedule (i.e., late arrival times), and the airport's resources (assuming all planes will need the same set of services). An episode is a 24-hour time period including the initial, nominal flight schedule. Another possible addition to the generative model are failure rates for the servicing resources. Because the nominal flight schedule for the airport is fixed ahead of time, generating an episode mostly concerns variations in arrival time (usually delays) and equipment breakdowns. Each arriving flight spawns a task group to service that flight.

Figure 4 shows the objective task structure for a small episode with two gates, two fuel trucks, and two baggage trucks. Gate 2 has an underground fuel tank, and thus enables local fuel delivery without needed a fuel truck. Slightly unusual relationships hold between using a gate and servicing the plane, because the gate must be held for the entire time of servicing (as opposed to a strict sequencing like enables). The subjective information available to any agent is the same, except for the arrival times of task groups (flights) in the future, which are only tentative.

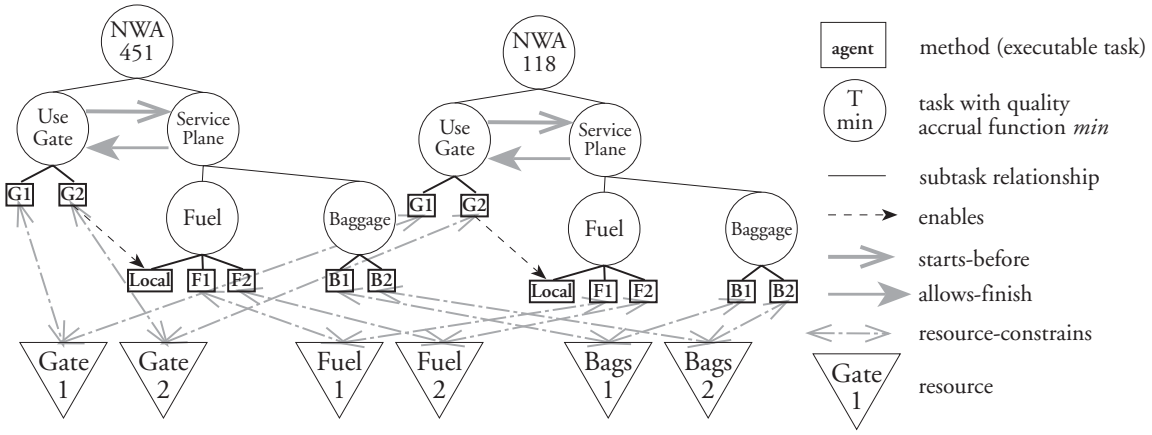

Figure 4: Objective task structure for a small airport resource management episode.

Notice the differences between this structure and the hospital structure. Resources are much more important here, and connect much more of the substructures to one another. Much of this comes out of the necessity to represent the effect of equipment travel times that are not represented in the hospital case (the point is not that 'equipment doesn't travel in the hospital' but that the travel times of the patients can be *standardized*—an important classical coordination technique). Travel times, however, cause every action taken by a resource (such as a fuel truck) to be related to every other action in a complex way, changing the effects at each potential successive action.

This strongly connected structure thus impacts the potential coordination mechanisms used. For example, in ARM, and in real airports, this scheduling is done in a centralized manner for a fixed set of resources (although one might consider resources such as baggage trucks as 'agents', in such a system these agents are slaves that only obey the commands of the central coordinator—similar to the bulldozers in Phoenix[4]).

Such a centralized scheduling process is obviously expensive and complex in terms of computation. The Dis-ARM project [29] looks to provide performance improvements over a centralized system by using multiple schedulers. The subjective view of the problem for each scheduler is considerably simplified by assigning each scheduler exclusive control of certain resources (for example, each scheduler could get one airport concourse of gates and a commensurate number of fuel and baggage trucks). Each scheduler tries to locally schedule the services for planes arriving in its concourse. The problem at each agent then becomes considerably simpler because it has been separated from the similar problems at other agents' concourses, and the agents can solve the problems in parallel as well. The downside is that the totally separate solutions are potentially more wasteful of resources. To combat this, the Dis-ARM system allows agents to lend resources to one another—note however that each lending act will connect initially unrelated task groups through resource relationships and make each agent's scheduling that much harder.

Other solutions are also suggested by the task structure. For example, since the services provided to each plane are already standardized, one could schedule the services at a constant headway (or one keyed to the business pattern, i.e., closer headway during the morning and evening rush times). Thus pre-scheduled and fixed slots would be established, and when a plane arrives (or soon before) it would be assigned to the next available slot (and associated gate). This technique removes the uncertainty in plane arrivals from the system (but is probably not used by airports because they wish for gate assignments to be made as early as possible, both for customers and for automated baggage handling). Another solution would be similar to the Dis-ARM structure; basically the corporate division organizational form—rather than direct negotiation, each scheduler (division) would apply for resources through a centralized 'front office' (which in turn centralizes all the rapidly changing information about the resources). A modification to this structure is a matrix organization where separate agents keep track of each class of resources (i.e., a baggage truck agent, a fuel truck agent, etc.) and then other agents are responsible for servicing the flights, drawing resources from the resource-class pools [41].

3.3 Internet Information Gathering

Another task environment mentioned briefly in the overview of this thesis was distributed information gathering. The Internet, as well as several popular commercial services, grow by leaps and bounds daily, providing rich and varied sources of information. Quickly and efficiently retrieving this information can be viewed as a set of interrelated tasks in an uncertain and dynamic environment. The same piece of data can be available via many different methods, and at many different locations; at any point of time only some of those locations are acessable through some subset of methods. It is extremely common nowadays for one to know precisely where a certain answer can be found but for one to have to undertake a time-consuming sequential search to find a currently available resource and access method. Within a single query, multiple agents could search in multiple locations in parallel and coordinate their

actions when it was useful. For example, the results of work by one agent may suggest the need for some of the existing agents to gather additional information, or it might suggest the need for a new division of tasks among the agents [30].

Generative level information in this task environment includes the frequencies and types of queries (are the agents beholden to individuals, or are they available to the network, which would result in very different usage patterns). It also includes information about the dynamics of the environment: how frequently do connections or links to physical information sources change, how often are the physical sources available, how often does the information on those sources change? How often are new physical sources being added?

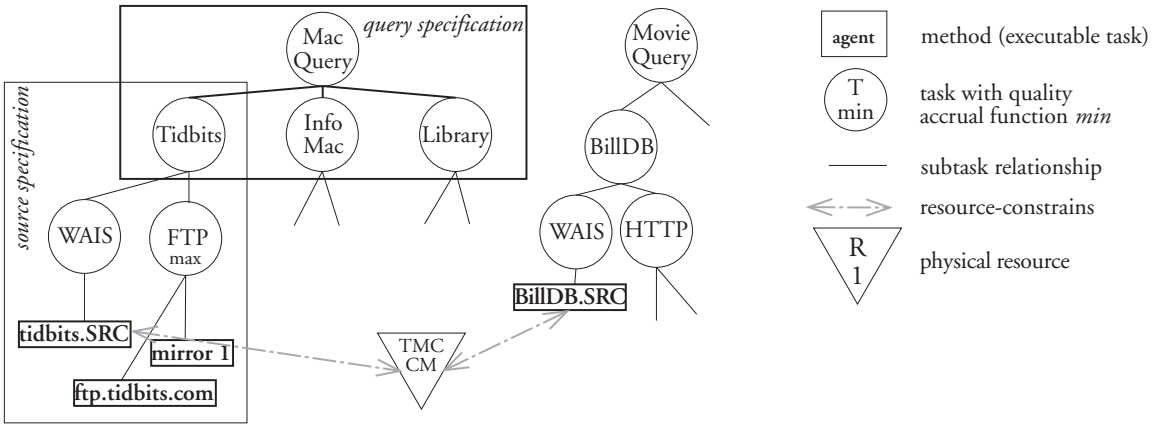

Figure 5: High-level, objective task structure for a two independent queries that resolve at one point to a single machine.

Figure 5 shows part of an objective level description of an example episode. Here two queries are being made—one about Macintosh information and one about movies. The Mac query might encompass several general sources of Mac information—the TidBits electronic newsletter, Info-Mac and other electronic file archives, Usenet newsgroups, and even standard library article searches. This particular query might be constructed by the interaction of a human user and a persistent Mac query intelligent agent. The task of searching each information source can be broken down into the different possible access methods for that source (e.g., WAIS, FTP, HTTP, a special-purpose agent that can use telnet to access a class of library databases, etc.). Each access method may still imply multiple physical resources (for example TidBits back issues can be found both at the mother site ftp.tidbits.com and at various *mirrors* of that site. Work done retrieving from mirror sites is redundant in the TÆMS sense—but which sites are likely to be operating at all? How fast will they be? Are they accepting connections?[8] One can imagine the enforcement of user performance criteria on the agents carrying out these tasks. One user might want a (not necessarily complete) answer immediately, another might be willing to wait for the agents to find the 'best' answer they can find. Queries might be continuous 'profiles' for information to be retrieved now and in the future as it becomes available.

Another point to note about Figure 5 is the presence of shared resources linking otherwise unrelated queries. In this example the shared physical resource is Thinking Machines' CM5

[8] These are all generative level questions.

WAIS server, which stores many example WAIS databases. Other similar examples include popular mirror and archive sites like uunet.uu.net and wuarchive.wustl.edu.

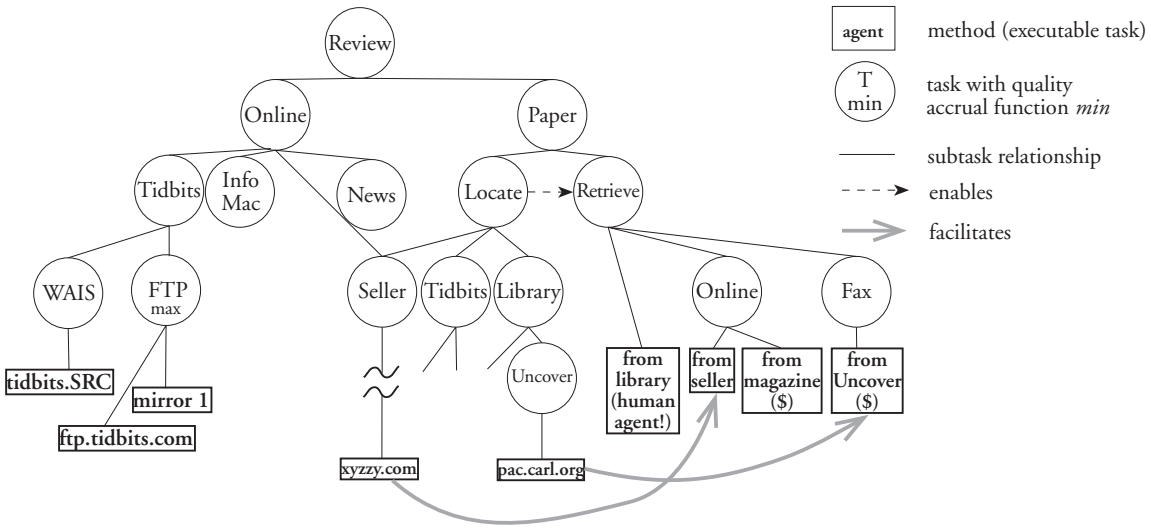

Figure 6: Mid-level, objective task structure for a single query for a review of a Macintosh product showing intra-query relationships.

Figure 6 shows more detail from a particular Macintosh query episode for a review of some Mac software product. Such reviews can be found both in online forms and in what were originally published paper forms (but which may now be online as well). A user may perhaps view truly published product information as being of higher quality. Online product information might be found in the review portion of the TidBits newsletter, or in collected product information in the Info-Mac archives, or in exchanges about a product in Usenet News. Finding paper reviews of a product can be reduced to locating a citation and then the actual article itself. TidBits contains the table of contents of the popular Macintosh magazines; some magazines are indexed by free periodical indices such as Uncover; sometimes the seller of a product will both have information available on the net and citations of reviews of their product. Once a citation is found, retrieval can be by hand or from online sources (but almost all of these will cost money). Note that some parts of this structure facilitate others, e.g., having used Uncover to come up with a citation it is very easy to have the article faxed to you at a price.

The task structure of this environment lends itself to a great deal of parallelism since a lot of work can be done in parallel and the interrelationships are relatively few and fixed. On the other hand, efficient planning for search according to a users preferences (e.g., quick response? best answer?) make the coordination problem more one of intelligent task decomposition.

3.4 The Pilot's Associate

The global coherence problems we would like to address occur in many systems other than the DVMT, such as the Pilot's Associate (PA) system [39], where situations occur that cause potentially complex and dynamically changing coordination relationships to appear between

goals that are spread over several agents. Each agent in the Pilot's Associate system has subtasks that other agents must fulfill, and receives tasks from other agents that only it can fulfill.

For example, assume that we are in a tactical situation, so the tactical planner is in control (see Figure 7). It has two ordered subtasks: turn on the active sensors (request to situation assessment), and get a detailed route of the plane's movements during the tactical maneuver (request to the mission planner). Turning on active sensors causes a plane to become a transmitter, and thus become easily detected (most of the time the plane uses passive sensors). Since this is dangerous, the situation assessment agent will ask the pilot-vehicle interface (PVI) to ask for pilot confirmation of the use of active sensors. The pilot, upon seeing the request, asks the PVI to plot the escape route of the plane on the screen in case things go wrong. The PVI passes this task to the mission planner.

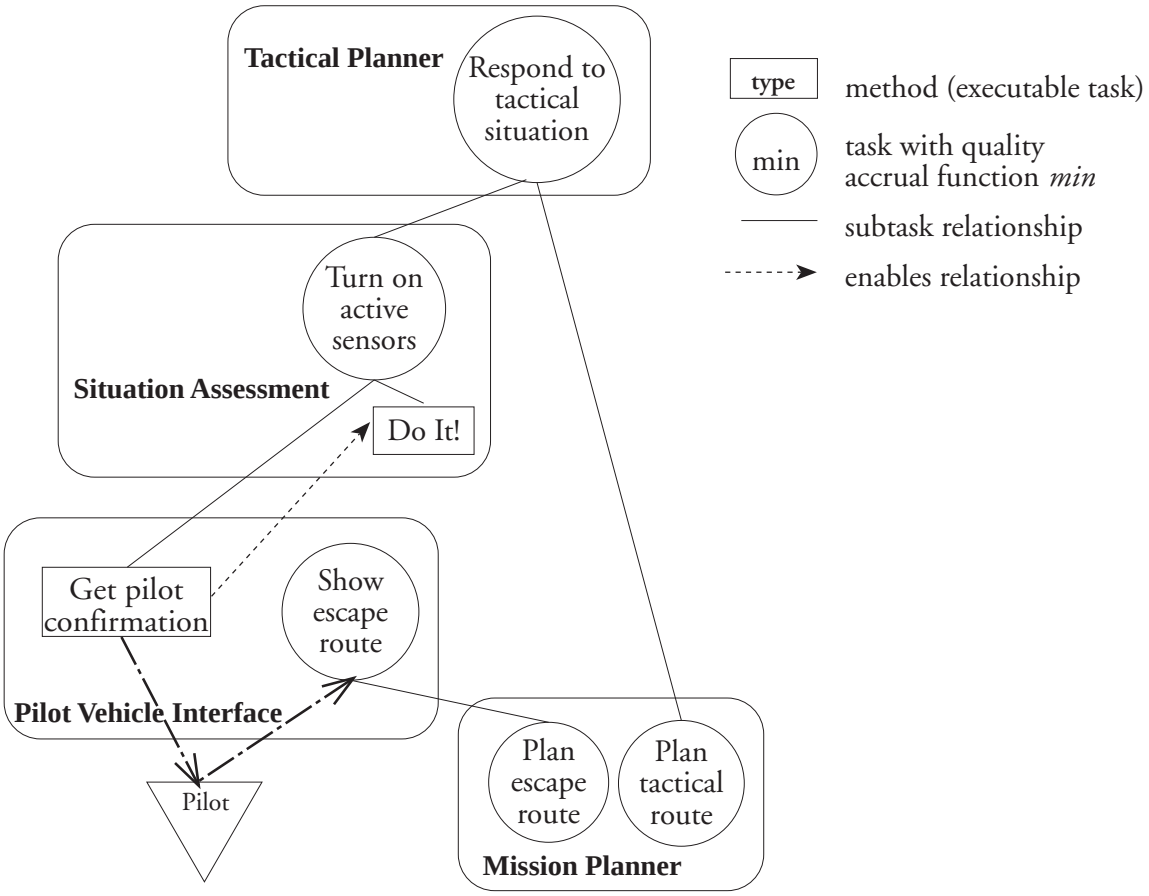

Figure 7: Dynamic Situations in Pilot's Associate. All tasks accrue quality with Min (AND).

Meanwhile, the tactical planner has asked the mission planner to produce the detailed route for the tactical maneuver. Which task does the mission planner act on first? From a local view, it may perhaps do the tactical planner request first because the tactical planner tasks are a high priority. But from a global perspective, we see that unless the mission planner plans the escape route, which is needed by the pilot in order to authorize turning on the active sensors, which is needed for the tactical planner to do its job, the whole system performance goal of handling the tactical situation is in jeopardy. Hence the mission planner should do the escape route plan

first.

Although I do not describe Generalized Partial Global Planning (GPGP) in detail here, let me briefly describe this scenario in those terms. There will be an overall deadline on the 'respond to tactical situation' task, which will be inherited by the 'plan tactical route' subtask. The enables relation between 'get pilot confirmation' and 'do it' (pinging the active sensors) could trigger a GPGP coordination mechanism to place an earlier deadline on the 'get pilot confirmation' task when it is communicated to the Pilot Vehicle Interface. The task 'show escape route', based on this, would also have an earlier deadline, and thus 'plan escape route' would have an earlier deadline (earlier than 'plan tactical route') as well. Thus the Mission Planner obtains the necessary information with which to schedule the two tasks. If the time to do both tasks is more than the earlier deadline and less than the later deadline, then the scheduler can only rationally choose to do 'plan escape route' first.

4 Support for Simulation

In order to test coordination and scheduling algorithms in general environments, the TÆMS simulator has a random task structure generator. This generator does *not* generate structures from a "real" application environment; the structures it generates are quite abstract. When working with TÆMS on a particular application such as those described above, we do not use this random generator, but rather one crafted to match the specific environment. Of course components of the random task structure generator can be reused when building environment-specific generators. The random, abstract environments are useful for general experimentation and performance analysis of coordination mechanisms and organizations.

The random environment generator has three parts: specifying general environmental parameters such as the inter-agent communication delay, specifying a generative subjective template, and specifying the potential classes of task groups (generative objective templates).

The default generative subjective template takes a list of agent names and a simple probability p_{overlaps} for the chance of overlapping methods. For each objective method generated and for each other agent, there is a p_{overlaps} chance that the agent will have an overlapping method. If there is no overlapping method, then the method is unique to a single agent's capabilities or role. All objective values are passed unchanged to the agents. Agents who execute information-gathering actions will receive information on all methods executable by them, and all parents of those methods, recursively. The default generative subjective template guarantees no other subjective properties. The level of expressiveness of the subjective model is where much of our current work on TÆMS is occurring—adding both organizational roles and better uncertainty representations.

A random environment will consist of one generative subjective template and a list of generative objective templates (one for each potential class of task groups). Each generative objective template takes parameters that indicate the average depth, branching factor, duration, max-quality, quality accumulation function distribution, deadline tightness, inter-arrival-time, and how redundant work should be valued.

The TÆMS random structure generator also allows the specifications of certain 'patterns' of relationships on top of this basic structure. These patterns have parameters that are in addition to the generative objective template parameters just mentioned. Some patterns allow

the expression of hard and soft task interrelationships (NLEs), the presence of cleanup methods (a single method that must be done to complete a task group), and the presence of fast-fallback methods (versions of regular methods that take less time but produce a lower quality).

5 Current Work

The TÆMS simulator has been used for several projects, some of them ongoing. Besides the verification of the analytical model of the distributed sensor network environments such as the DVMT [8, 9], researchers at UMass have used TÆMS to develop new scheduling algorithms [15] that help a computational agent to choose what actions to take, in what order to take them, and when. Ongoing work includes the development of scheduling algorithms for scheduling the work of groups of agents.

Another area of work has been the development and evaluation of the Generalized Partial Global Planning (GPGP) family of coordination algorithms. GPGP both generalizes and extends Durfee's Partial Global Planning (PGP) algorithm [13]. Our approach has several unique features:

- Each mechanism is defined as a response to certain features in the current subjective task environment. Each mechanism can be removed entirely, or parameterized so that it is only active for some portion of an episode. New mechanisms can be defined; an initial set of five mechanisms has been examined. These are "micro" mechanisms when compared to, for example, Mintzberg's "macro" mechanisms [28], but they too can be clustered into archetypical forms [6].

- GPGP works in conjunction with an existing agent architecture and local scheduler. My experimental results were achieved using a 'design-to-time' soft real-time local scheduler developed by Garvey [16]. GPGP thus makes assumptions about the internal architecture of an agent (unlike TÆMS). In the GPGP approach, an agent's local scheduling component develops several possible courses of actions (the schedules) and an agent's decision-making component chooses one course to follow for some period of time. The GPGP coordination component, then, works by supplying information and constraints to the local scheduler, and to other agent's coordination components. The most common constraint is a *commitment* to achieve a certain level of quality at a task by a certain time.

- GPGP, unlike PGP, is not tied to a single domain. Secondly, GPGP allows more agent heterogeneity than PGP with respect to agent capabilities. Finally, GPGP mechanisms in general exchange less information than the PGP algorithm, and the information that GPGP mechanisms exchange can be at different levels of abstraction. PGP agents generally communicate complete schedules at a single, fixed level of abstraction. GPGP mechanisms communicate scheduling commitments to particular tasks, at any convenient level of abstraction.

An example of a GPGP coordination mechanism is one that handles simple method redundancy. If more than one agent has an otherwise equivalent method for accomplishing a task, then an agent that schedules such a method will commit to executing it, and will notify the other agents of its commitment. If more than one agent should happen to commit to

a redundant method, the mechanism takes care of retracting all but one of the redundant commitments. The fact that most of the GPGP coordination mechanisms use commitments to other agents as local scheduling constraints is the reason that the GPGP family of algorithms requires cooperative agents. Nothing in TÆMS the underlying task structure representation, requires agents to be either cooperative, antagonistic, or simply self-motivated. Details about GPGP and evaluations of the mechanisms can be found elsewhere [6, 11].

6 A Simulation Example

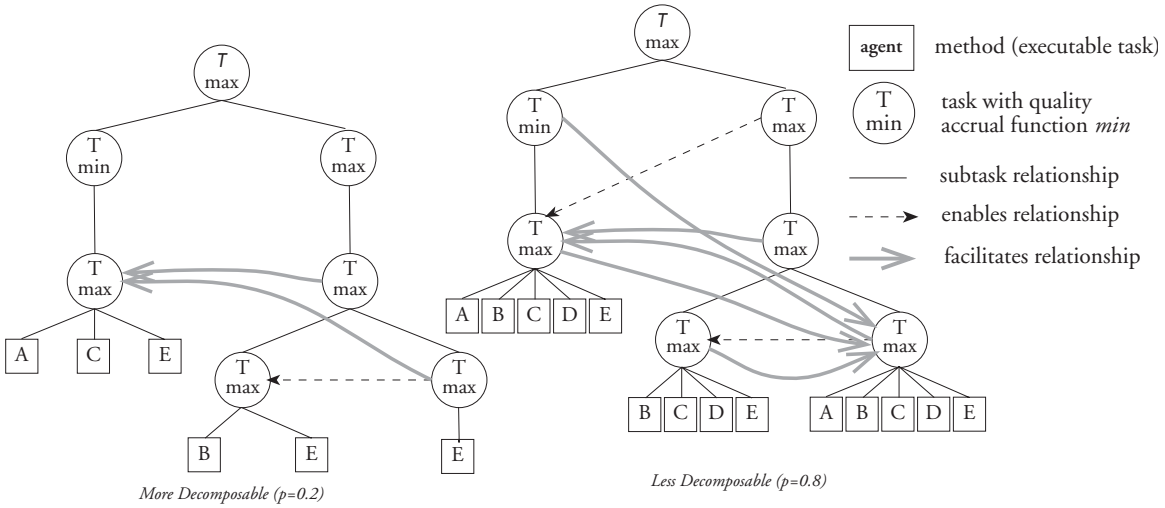

Figure 8: Example of a randomly generated objective task structure

,

Let's look at a small example (from [10]) in which we use TÆMS as a simulator to explore hypotheses about the interactions between environmental and agent-structural characteristics. We use as an example a question explored by Burton and Obel: is there a significant difference in performance due to either the choice of organizational structure or the decomposability of technology[9]?

We equate a technology with a TÆMS task structure, instead of a linear program. Task structures allow us to use a clear interval measure for decomposability, namely the probability of a task interrelationship (in this example enables, facilitates, and overlaps). We define a nearly decomposable task structure to have a base probability of 0.2 for these three coordination relationships and a less decomposable task structure to have a base probability of 0.8 (see Figure 8).

Burton and Obel were exploring the difference in M-form (multidivisional) and U-form (unitary–functional) hierarchical structures; here we will analyze the GPGP family of team-oriented coordination algorithms. For our structural variable we will vary the communication of non-local views (just one of the GPGP coordination mechanisms). Informally, we will

[9] *Technology* is used here in the management science sense of "the physical method by which resources are converted into products or services" or a "means for doing work" [2, 34].

be contrasting the situation where each agent makes commitments and communicates results based only on local information (*no* non-local view) with one where the agents freely share task structure information with one another precisely when there are coordination relationships (*partial* non-local view). Figure 9 shows an example—note that in neither case does the agent have the *global* view of Figure 8.

Burton and Obel used a profitability index as their performance measure, derived from the percentage of optimal profit achieved. In general, the scheduling an arbitrary TÆMS task structure is an NP-hard problem and so we do not have access to optimal solutions. Instead we compare performance directly on four scales: the number of communication actions, the amount of time spent executing methods, the final quality achieved, and the termination time. Simulation runs for each of the four combinations of non-local view policy and level of task decomposability were done in matched sets—the randomly generated episode was the same for each combination with the exception of more coordination relationships (including more overlapping methods) being added in the less decomposable task structures. Following Burton and Obel, we used the non-parametric Friedman two-way analysis of variance by ranks test for our hypotheses. The assumptions of this test are that each block (in our case, randomly generated episode) is independent of the others and that each block contains matched observations that may be ranked with respect to one another. The null hypothesis is that the populations within each block are identical.

Parameter	**Value**
Mean Branching factor (Poisson)	1
Mean Depth (Poisson)	3
Mean Duration (exponential)	10
Redundant Method QAF	Max
Number of task groups	1
Task QAF distribution	(50% min) (50% max)
Decomposition parameter	p = 0.2 or 0.8
Hard CR distribution	(p enables) ((1-p) none)
Soft CR distribution	(p facilitates) (10% hinders) ((.9-p) none)
Chance of overlaps (binomial)	p

Table 1: Parameters used to generate the 40 random episodes

We generated 40 random episodes of a single task group, each episode was replicated for the four combinations in each block. We used teams consisting of 5 agents; the other parameters used in generating the task structures are summarized in Table 1 and a typical randomly generated structure is shown in Figure 8. Figure 9 shows the difference in the local view of one agent with and without creating partial non-local views. We first tested two major hypotheses:

Hypothesis 1: *There is no difference in performance between agents with a partial non-local view and those without.* For the communication and method execution performance measures, we reject the null hypothesis at the 0.001 level. We cannot reject the null hypothesis that there is no difference in final quality and termination time. Teams of computational

agents that exchange information about their private, local views consistently exchange more messages (in this experiment, a mean increase of 7 messages) but do less work (here, a mean decrease of 20 time units of work, probably due mostly to avoiding redundancy).

Hypothesis 2: *There is no difference in performance due to the level of decomposability of technology.* For the communication and method execution performance measures, we reject the null hypothesis at the 0.001 level. We cannot reject the null hypothesis that there is no difference in final quality and termination time. Teams of computational agents, regardless of their policy on the exchange of private, local information communicate more messages (in this experiment, a mean increase of 47 messages) and do more work (here, a mean increase of 24 time units) when faced with less decomposable computational task structures (technology).

Again following Burton and Obel, we next test for interaction effects between non-local view policy and level of technology decomposability by calculating the differences in performance at each level of decomposability, and then testing across non-local view policy. This test was significant at the 0.05 level for communication, meaning that the difference in the amount of communication under the two policies is itself different depending on whether task decomposability is high or low. This difference however is small (two communication actions in this experiment) and was not verified in a second independent test.

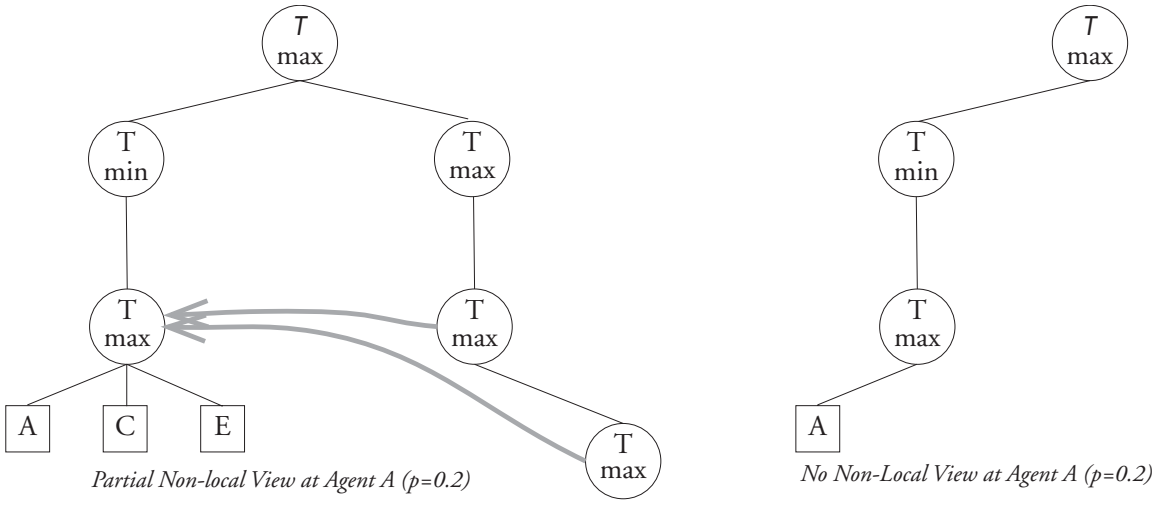

Figure 9: Example of the local view at Agent A when the team shares private information to create a partial non-local view and when it does not.

7 Conclusions and Future Work

In this chapter we have given an overview of the TÆMS (Task Analysis, Environment Modeling, and Simulation) framework with which to model and simulate complex, computationally intensive task environments at multiple levels of abstraction and from multiple viewpoints. TÆMS is a tool for building and testing computational theories of coordination and organization.

Such theories can relate environmental features (such as a large variance in workload) to useful behaviors (load balancing meta-level communication [8]). A TÆMS model provides information about the structure and other characteristics of tasks in an environment, and about what task information is known and what actions can be taken by agents in certain organizational roles or with certain capabilities. TÆMS models of task environments are compatible with many popular models of agents' internal structure—evaluating a computational theory requires putting these two parts together during the simulation.

So far, TÆMS has been used primarily to computationally operationalize various aspects of textual organization theory and examine its impact on the design of distributed computing systems. GPGP derives some of its structure from elementary contingency theory; we are looking into a computational version of Williamson's transaction cost economics theories. We intend to apply these results to the organization of a multi-agent, distributed information gathering system. Here transaction-specific investments are not physical equipment but costly meta-information about network information sources and alternatives. Other work is focusing on extending the set of GPGP coordination mechanisms, and studying how agent organizations can learn to adapt their coordination mechanisms to new task environments.

References

[1] Mark Boddy and Thomas Dean. Solving time-dependent planning problems. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 979–984, Detroit, August 1989.

[2] Richard M. Burton and Børge Obel. *Designing Efficient Organizations: Modelling and Experimentation*. North Holland, Amsterdam, 1984.

[3] K.M. Carley and M.J. Prietula. ACTS theory: Extending the model of bounded rationality. In K.M. Carley and M.J. Prietula, editors, *Computational Organization Theory*, chapter 4, pages 55–89. Lawrence Erlbaum Associates, Hillsdale, NJ, 1994.

[4] Paul Cohen, Michael Greenberg, David Hart, and Adele Howe. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3):33–48, Fall 1989. Also COINS-TR-89-61.

[5] Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(3):213–261, 1990.

[6] Keith S. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms*. PhD thesis, University of Massachusetts, 1995.

[7] Keith S. Decker and Victor R. Lesser. Analyzing a quantitative coordination relationship. *Group Decision and Negotiation*, 2(3):195–217, 1993.

[8] Keith S. Decker and Victor R. Lesser. An approach to analyzing the need for meta-level communication. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 360–366, Chambéry, France, August 1993.

[9] Keith S. Decker and Victor R. Lesser. A one-shot dynamic coordination algorithm for distributed sensor networks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 210–216, Washington, July 1993.

[10] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex environments. *International Journal of Intelligent Systems in Accounting, Finance, and Management*, 2(4):215–234, December 1993. Special issue on "Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior".

[11] Keith S. Decker and Victor R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems*, San Francisco, June 1995. AAAI Press.

[12] Edmund H. Durfee, Victor R. Lesser, and Daniel D. Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 36(11):1275–1291, November 1987.

[13] 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.

[14] J. Galbraith. *Organizational Design*. Addison-Wesley, Reading, MA, 1977.

[15] Alan Garvey, Marty Humphrey, and Victor Lesser. Task interdependencies in design-to-time real-time scheduling. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 580–585, Washington, July 1993.

[16] Alan Garvey and Victor Lesser. Design-to-time real-time scheduling. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(6):1491–1502, 1993.

[17] Les Gasser. Social conceptions of knowledge and action. *Artificial Intelligence*, 47(1):107–138, 1991.

[18] Piotr J. Gmytrasiewicz, Edmund H. Durfee, and David K. Wehe. A decision-theoretic approach to coordinating multiagent interactions. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 62–68, Sydney, Australia, August 1991.

[19] David W. Hildum. *Flexibility in a Knowledge-Based System for Solving Dynamic Resource-Constrained Scheduling Problems*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, September 1994.

[20] Y.-C. Ho. Team decsision theory and information structures. *Proceedings of the IEEE*, 68, June 1980.

[21] Eric J. Horvitz. Reasoning under varying and uncertain resource constraints. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, August 1988.

[22] Paul Lawrence and Jay Lorsch. *Organization and Environment*. Harvard University Press, Cambridge, MA, 1967.

[23] R.E. Levitt, P.G. Cohen, J.C. Kunz, C. Nass, T. Christiansen, and Y. Jin. The virtual design team: Simulating how organizational structure and communication tools affect team performance. In K.M. Carley and M.J. Prietula, editors, *Computational Organization Theory*. Lawrence Erlbaum Associates, 1994.

[24] Z. Lin. A theoretical evaluation of measures of organizational design: Interrelationship and performance predictability. In K.M. Carley and M.J. Prietula, editors, *Computational Organization Theory*, chapter 6, pages 113–159. Lawrence Erlbaum Associates, Hillsdale, NJ, 1994.

[25] Z. Lin and K. Carley. Proactive or reactive: An analysis of the effect of agent style on organizational decision-making performance. *International Journal of Intelligent Systems in Accounting, Finance, and Management*, 2(4):271–289, December 1993. Special issue on "Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior".

[26] Thomas W. Malone. Modeling coordination in organizations and markets. *Management Science*, 33:1317–1332, 1987.

[27] J. G. March and H. A. Simon. *Organizations*. Wiley, New York, 1958.

[28] H. Mintzberg. *Structure in fives: designing effective organizations*. Prentice-Hall, Englewood Cliffs, N.J., 1983.

[29] D.E. 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, August 1994.

[30] Tim Oates, M. V. Nagendra Prasad, and Victor R. Lesser. Cooperative information gathering: A distributed problem solving approach. Technical Report 94-66, Department of Computer Science, University of Massachusetts, September 1994.

[31] P. S. Ow, M. J. Prietula, and W. Hsu. Configuring knowledge-based systems to organizational structures: Issues and examples in multiple agent support. In L. F. Pau, J. Motiwalla, Y. H. Pao, and H. H. Teh, editors, *Expert Systems in Economics, Banking, and Management*, pages 309–318. North-Holland, Amsterdam, 1989.

[32] J. S. Rosenschein and M. R. Genesereth. Deals among rational agents. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 91–99, August 1985.

[33] Stuart J. Russell and Shlomo Zilberstein. Composing real-time systems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 212–217, Sydney, Australia, August 1991.

[34] W. Richard Scott. *Organizations: Rational, Natural, and Open Systems*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.

[35] S. Sen and E. Durfee. On the design of an adaptive meeting scheduler. In *Proceedings IEEE Conference on AI Applications*, 1994.

[36] Yoav Shoham. AGENT0: A simple agent language and its interpreter. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 704–709, Anaheim, July 1991.

[37] Herbert A. Simon. *Models of Man*. Wiley, New York, 1957.

[38] Herbert A. Simon. *Models of Bounded Rationality, Volume 2*. The MIT Press, Cambridge, MA, 1982.

[39] David Smith and Martin Broadwell. Plan coordination in support of expert systems integration. In *Proceedings of the DARPA Knowledge-Based Planning Workshop*, pages 12.1–12.6, December 1987.

[40] Arthur L. Stinchcombe. *Information and Organizations*. University of California Press, Berkeley, CA, 1990.

[41] K. Sycara, S. Roth, N. Sadeh, and M. Fox. Distributed constrained heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1446–1461, November/December 1991.

[42] James D. Thompson. *Organizations in Action*. McGraw-Hill, New York, 1967.

[43] Frank v. Martial. *Coordinating Plans of Autonomous Agents*. Springer-Verlag, Berlin, 1992. Lecture Notes in Artificial Intelligence no. 610.

[44] D.L. Westbrook, S.D. Anderson, D.M. Hart, and P.R. Cohen. Common lisp instrumentation package: User manual. Technical Report 94–26, Department of Computer Science, University of Massachusetts, 1994.

[45] G. Zlotkin and J. S. Rosenschein. Incomplete information and deception in multi-agent negotiation. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 225–231, Sydney, Australia, August 1991.