Evolution of the GPGP Domain-Independent Coordination Framework[1]

by

V. Lesser, K. Decker, N. Carver, A. Garvey, D. Neiman,
M. Nagendra Prasad and T. Wagner

University of Massachusetts
Computer Science Technical Report 1998-05
January 1998

Abstract:

The GPGP domain-independent coordination structure for small agent groups was first detailed in an ICMAS95 paper. In this paper, we discuss the evolution of this framework over the last three years based on its being applied to a number of applications involving distributed situation assessment, information gathering and management, coordination of concurrent engineering activities and hospital scheduling. First, we review the basic architecture of GPGP and then present extensions to the TAEMS domain-independent representation of agent activities. We next describe new coordination mechanisms for use in resource sharing and contracting and the need for more complex coordination mechanisms. Finally, extensions to GPGP that permit the description and learning of situation-specific coordination strategies are detailed along with techniques for using GPGP in large agent organizations.

---

[1] 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), and the National Science Foundation (under Grant No. IRI-9523419). The views and conclusions contained herein are those of the authors 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.

Evolution of the GPGP Domain-Independent Coordination Framework[2]

V. Lesser, K. Decker[3], N. Carver[4], A. Garvey[5], D. Neiman, M. Nagendra Prasad[6] and T. Wagner

Computer Science Department
University of Massachusetts, Amherst

Abstract:

The GPGP domain-independent coordination structure for small agent groups was first detailed in an ICMAS95 paper. In this paper, we discuss the evolution of this framework over the last three years based on its being applied to a number of applications involving distributed situation assessment, information gathering and management, coordination of concurrent engineering activities and hospital scheduling. First, we review the basic architecture of GPGP and then present extensions to the TAEMS domain-independent representation of agent activities. We next describe new coordination mechanisms for use in resource sharing and contracting and the need for more complex coordination mechanisms. Finally, extensions to GPGP that permit the description and learning of situation-specific coordination strategies are detailed along with techniques for using GPGP in large agent organizations.

Topics: Coordination

---

[2] 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), and the National Science Foundation (under Grant No. IRI-9523419). The views and conclusions contained herein are those of the authors 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.

[3] At Computer and Information Science Dept., University of Delaware.
[4] At Computer Science Dept., Southern Illinois University.
[5] At Computer Science Dept., Truman State University.
[6] At Center for Strategic Technology Research, Andersen Consulting.

Introduction

Generalized Partial Global Planning (GPGP) was developed as a domain-independent framework for coordinating the real-time activities of small teams of agents [1, 5]. Over the last three years, we have significantly extended the framework based on our experience applying it to a number of applications involving distributed situation assessment, information gathering and management, coordination of concurrent engineering activities, and hospital scheduling [2, 3, 9, 10]. The focus of this paper is to discuss these extensions and how they permit us to implement coordination strategies for a wide range of multi-agent systems. We start out by first reviewing the basic concepts behind GPGP and how this framework relates to other approaches.

Review of Basic Architecture

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 is not entirely local (i.e., 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 affecting the tasks an agent will execute, when they will be executed, and where their results will be transmitted. Integral to GPGP is domain-independent scheduler that, based on commitments, agents' goals and other agent activity constraints, creates a schedule of activities for the agent to perform. In this way, GPGP coordinates the activity of agents through *modulating their* local control as a result of placing constraints and commitments on the local scheduler. No one 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), we can create a wide set of different coordination responses.

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, (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*, (c) a precise, quantitative definition of the degree of achievement in terms of measurable characteristics such as solution quality and time, (d) task relationships that indicate how basic actions or abstract task

achievement effect task characteristics (e.g., quality and, time) elsewhere in the task structure[7]. The language that we use to represent these task structures is called TAEMS [1, 4, 15]. This choice of representation is quite unique and powerful. Unlike many systems, we deal with *worth-oriented* [13] environments where a goal is not black and white, but rather has a degree of achievement associated with it. We represent this in TAEMS by tracking a quantitative vector of task characteristics or criteria over which some utility preference may be expressed [18]. We also allow 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.

In terms of BDI style architectures [12], an agent selects from among its desires an intended set which are then "planned" for, and whose component actions are then "scheduled" for execution. In this paper, we view "planning" as the process of elucidating a task structure with a manageable number of possibilities, and "scheduling" as the process of selecting specific intended actions and locating them in time. GPGP, despite the name, has so far primarily focused on coordinating the scheduling process[8].

The GPGP approach uses this basic TAEMS task structure representation, and adds two important extensions: partial representations of the task structures at other agents, and local and non-local commitments to task achievement. Each GPGP coordination mechanism is specified with respect to features in these dynamically changing task structures, and so provides flexible coordinated responses across different problem domains. The quantitative, worth-oriented approach has several benefits when compared to symbolic, preplanned, domain-specific approaches [7]. In particular, GPGP is not domain-specific, but rather task-structure-characteristic-specific. The same task relationships may appear across many different domains, and the same coordination mechanisms can be used[9]. Efficient and effective coordination must account for the benefits and the costs of coordination in the current situation. The current situation includes the goals that the agent is currently pursuing, the goals it will likely pursue in the near term, the characteristics of the abstract tasks and basic actions available to achieve these goals, their relationships to other tasks (possibly at other agents), and the degree of achievement necessary for each goal. Purely symbolic reasoning

[7] Hard relationships (e.g., *enables*) denote when the result from one problem-solving activity is required to perform another, or when performing one activity precludes the performance of another. Soft relationships, such as *facilitates*, expresses the notion that the results of one activity may be beneficial (or harmful) to another activity, but that the results are not required in order to perform the activity.

[8] The name GPGP derives from Durfee's Partial Global Planning [6], which by today's definition of AI planning also focuses on the scheduling end of the planning-scheduling spectrum.

[9] The question of which mechanism to use when more than one is possible is a question about organizational/societal norms.

about costs and benefits can be extremely complex, particularly in large systems and open environments. Thus, the ability to reason quantitatively about the 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 [4].

The initial set of mechanisms for GPGP were derived from Durfee's work on PGP [6] in the Distributed Vehicle Monitoring Testbed (DVMT). These five mechanisms were: (1) communicate non-local views; (2) communicate appropriate results; (3) avoid redundancy; and (4,5) handle hard and soft coordination relationships that extend between tasks at two agents [1]. These five basic mechanisms formed a domain-independent set for basic agent teamwork. As such, it is interesting to compare them to Tambe's more recent work on flexible teamwork [14], in turn based on Cohen & Levesque's work [8]. In particular, the five initial GPGP mechanisms do not separate the environment-centered coordination actions from actions oriented toward teamwork itself. Also, GPGP currently only represents joint intentions *implicitly* by communicating parts of one agent's local task structure to another agent.

A striking example is the Cohen and Levesque model's prediction that team members should communicate when they believe that the object of a joint persistent goal has been achieved, found unachievable, or irrelevant. The GPGP "communicate results" mechanism contains a clause to communicate a "final result" (degrees of achievement) for every task group (implicit joint persistent goal) indicating that the agent believes no further actions can be done for that task group. Interestingly, this is one of the small generalizations that was added when creating the general GPGP mechanisms from the original DVMT PGP heuristics. The Tambe STEAM work thus points to a need to more carefully define coordination mechanisms to separate bottom-up, environment-centered coordination activities from top-down, organizationally-prescribed teamwork coordination activities—an exciting area of future work. On the other hand, the success of using GPGP environment-centered mechanisms in multiple environments demonstrates that there are indeed general mechanisms for solving domain-level coordination problems in different environments, because the task relationships in those environments can be characterized in a general way. While STEAM's teamwork rules are general, it still requires new, different, domain-specific coordination rules for each domain. Another difference is that TAEMS and GPGP deal in worth oriented environments, thus goals are achieved to varying degrees, rather than black and white "achieved or not achieved."

Extensions to TAEMS

In its original conceptualization, TAEMS was a hierarchical task representation language that featured the ability to express alternative ways of performing tasks, characterization of methods according to expected quality and expected duration, and the representation of interactions between tasks. As a result of using it in a number of applications, TAEMS has evolved considerably in representational power and in its range of applications.

As TAEMS has evolved, we have moved beyond simple quality/time trade-off reasoning to a multi-dimensional satisficing methodology [15]. Methods are no longer simply characterized in terms of expected quality and duration values but are characterized statistically via discrete probability distributions in three dimensions: quality, cost, and duration; and the work is extensible to multiple other dimensions as well. The switch from expected values to discrete distributions enables reasoning about the certainty [16] of particular courses of action, perhaps performing contingency analysis [11], as well as the quality/cost/time trade-offs. This more complete view is crucial in highly constrained situations (e.g., where deadlines are present). For example, in meeting an important commitment, it may be preferable to choose a highly certain course of action that has only moderate overall quality to ensure that results are available by the deadline, rather than a less certain solution path, even though it may have the possibility of a very high-quality payoff. The enhancement of multiple attribute dimensions, including uncertainty, is also important because the representation enables the multi-agent system designer to better characterize the desired system performance [15, 17]. This new performance characterization takes into account both hard and soft constraints on quality, cost and duration of the schedule of tasks intended to meet a high-level goal or commitment. Additionally, the amount of uncertainty that is tolerable in meeting these constraints can be specified [16].

In TAEMS, progress toward the problem-solving objective is expressed and tabulated in terms of quality. Tasks accumulate quality from their subtasks according to *quality accumulation functions (qafs)*. Originally, *qafs* defined only which combinations of subtasks could be performed to achieve the parent task and how the subtask qualities are tabulated at the parent task. For example, the *sum() qaf* denotes that the power set of the child tasks, minus the empty set, can be performed to achieve the parent task and that the parent task's quality is the sum of the children's qualities. In the current TAEMS, *qafs* may also define orderings. In all, we have developed a suite of 15 different *qafs* to represent the needs of different application domains.

Another important enhancement is the representation of different possible outcomes for primitive activities. For example, an action may produce a result of type A, type B, or it may fail entirely. With the outcome enhancement, clients can represent cases where a result of type A is beneficial to some other method, say method A; type B is beneficial to yet another method, method B; and the failure outcome enables a recovery action. While outcomes have always been included conceptually in TAEMS, either via entries in a method's quality, cost, and duration distributions or via a contribution to the expected values in the early versions of TAEMS, this new explicit outcome representation enables the expression of task interactions that are based on particular outcomes.

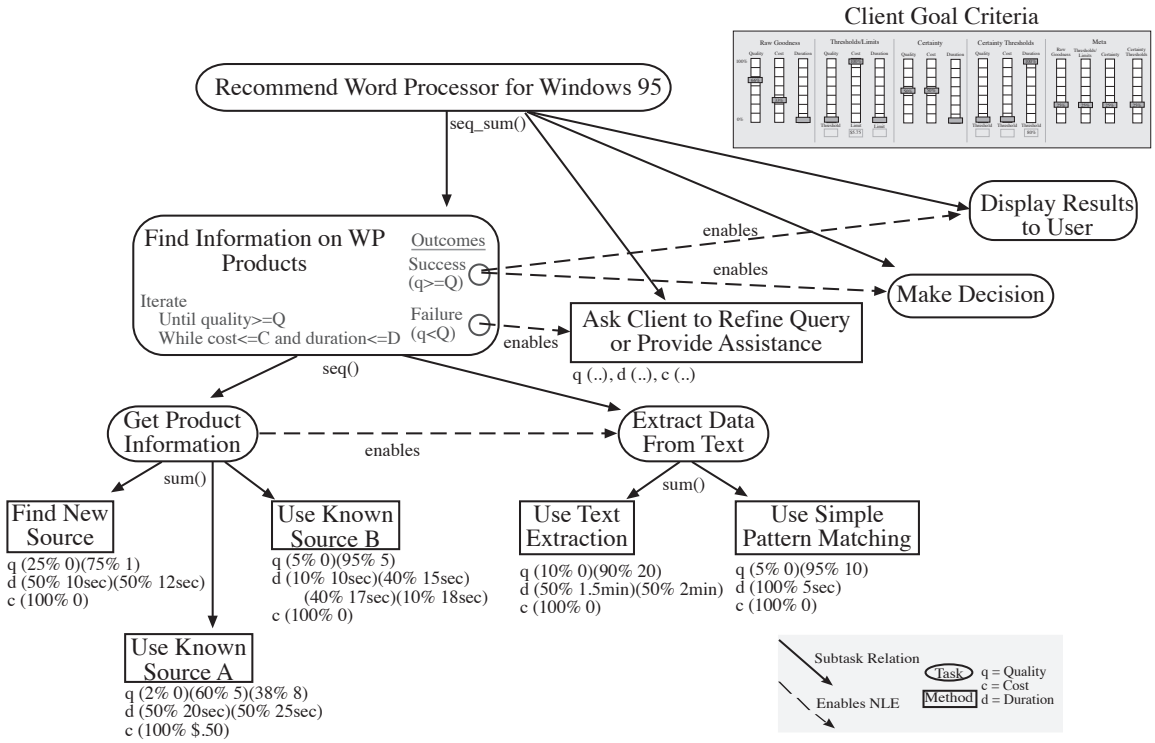

Figure 1: An example of a TAEMS task structure for an Information Gathering Agent

Figure 1 shows an example information gathering TAEMS task structure that contains some of these new extensions to TAEMS. The top level task is to recommend a word processor for Windows 95. It has four subtasks: one that pertains to finding information about various products; one for making the decision about which product to purchase; one that handles displaying the results to the client; and a recovery method that executes if the agent is unable to find enough information to support a recommendation. These four tasks are governed by a *seq_sum() qaf*, which indicates that the tasks must be performed in order and that the quality of the top level task is a sum of the qualities of these four subtasks. The task that handles finding and extracting usable

information about word processors, Find-Information-on-WP-Products, is an iterative task which means that the agent will loop over its subtasks until either the desired level of quality, Q, is reached or one of the time/cost limits is crossed. If Q is reached the outcome of the loop task is the success outcome, otherwise the outcome of the task is the failure outcome. If the outcome is success, the enables non-local-effects leading from the success outcome to the Make-Decision and Display-Results subtasks are satisfied and these other tasks may then be executed. In this case, the enables relation models the requirement that a certain level of product information is required to make a decision about which product to purchase and that to display results to the user, you must have something to display. However, if the outcome is failure, then these other tasks are not enabled and cannot be performed, but, the recovery method, Ask-Client-to-Refine-Query is enabled and may be executed. The Get-Product-Information task illustrates the notion of alternatives in TAEMS. This task has three subtasks, one or more of which may be performed to obtain product information. The subtasks have different quality, cost, and duration characteristics and the agent will determine which subtasks to execute depending on the current problem solving context. Similarly with the Extract-Data-from-Text task; there are two different ways to perform this task and the current problem solving context will determine which is appropriate.

The explicit representation of non-computational resources, such as network bandwidth, databases, and physical resources like printers, is another important TAEMS enhancement. To do this, we represent each resource explicitly as an object with state with relationships to tasks in the existing task structure. As with task-task relations, a task-resource relation indicates how the execution of a task affects the state of the resource, and how the state of the resource affects the characteristics (quality, cost, duration) of the tasks it is related to. Other enhancements include new *qafs* that more closely model certain domain activities and new task relationships. Iteration and repetitive tasks are still more enhancements soon to be added to TAEMS.

It is interesting to note that the explicit representation of alternative ways to achieve tasks, an original TAEMS idea, is still one of its most important features today. It is also interesting to note that one of the most fundamental shifts in TAEMS is actually in the way TAEMS is related to agent problem solving. Originally, we viewed TAEMS as a way to represent agent problem-solving activities, that is, agents would literally use TAEMS internally. However, we now view TAEMS as an abstract model of agent problem solving activities. Typically in our work, a domain problem solver uses its own internal representation for problem solving and this representation is abstracted into a TAEMS task structure for use by GPGP. This view motivates the continuous evolution of TAEMS. The addition of new features and new representations is caused by the constant tension between representational power and the ability to reason cheaply about an agent's problem-solving

activities. There is a trade-off between representing large amounts of problem-solving possibilities, and contingencies, and maintaining a somewhat wieldly modeling framework. For example, if the problem solver enumerates all possible problem-solving actions for every step in the problem-solving process, and all possible actions caused by carrying out the first set of actions, and all possible contingencies in the case of failure or a change in the environment, the TAEMS model becomes overly complex and unusable. However, through new constructs like iteration, outcomes, and *qafs* that impose orderings and other semantics, the representational power of TAEMS is increased without increasing the computational complexity of reasoning with the TAEMS models. Through outcomes, iteration, and the new *qafs*, the problem solver can model key contingencies, key repetitive actions, and thus provide GPGP with more information while not overloading it with details. Without these features, an agent might have to constantly communicate with GPGP about how its view of its future activities needs to be modified as the result of its incremental execution of its tasks; this modified view may entail significant re-coordination activities. Instead, GPGP can now, when it makes its original coordination decisions based on the agent's description of its activities, factor into these decisions that an agent's activities may change in certain ways as a result of the partial execution of its activities. For example, instead of establishing a commitment that is highly uncertain because there is a significant possibility that a task established to satisfy the commitment may not be completed successfully, it is now possible to analyze whether there would be another way that the agent could successfully complete the commitment with additional time. Thus, the coordination mechanism can, in making the original commitment, factor in the implications of the original task failing and an alternative method being required, which can result in making a different commitment that has a much higher certainty of being successfully honored.

Extension to GPGP

Situation-specific control

GPGP (Generalized Partial Global Planning) is a modularized approach to cooperative multi-agent coordination. GPGP's modularized design enables different modules, with different overhead costs, to be applied independently from one another. This provides agents with flexibility with respect to the computational and resource costs of coordination. For example, when resources are tight, agents may elect to use only the most rudimentary coordination mechanisms. When resources are less scarce, or the benefit of coordinating outweighs the cost of the coordination overhead, agents may coordinate over "optional" TAEMS task relationships, like the soft facilitates relationship. The GPGP modularized coordination approach facilitates a range of coordination cost/benefit options for any problem-solving episode as different subsets of coordination modules may be applied.

In recent work, we have significantly expanded on this idea of situation-specific coordination strategies. In [9], we showed that a suite of coordination mechanisms could be dynamically selected based on the agents exchanging meta-level information about their state in terms of measures, based on an agent's task structures, that indicated such things as their current loading, time pressure in terms of meeting deadlines, and the potential benefits of interaction with other agents. These same measures were also used by a learning mechanism which over time determined for a specific situation which was the most appropriate configuration of coordination mechanisms to use.

More complex view of commitments

We have also introduced the idea of coordination mechanisms that are based on default knowledge which allows us to represent coordination based on social laws. As previously discussed, coordination in GPGP is achieved through the use of *commitments*, that is, inter-agent contracts to perform certain tasks by certain times. The commitments which are dynamically constructed generally fall into three categories: 1) *deadline* commitments are contracts to perform work by a certain deadline; 2) *earliest start time* commitments are agreements to hold off on performing certain tasks until a certain time has passed; and 3) *do* commitments are agreements to perform certain tasks without a particular time constraint. By allowing these commitments to be specified *a priori* as part of the TAEMS task structure, 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 [9]. 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. An easy extension of this idea that we have not yet implemented is for either agent to notify the other that it either cannot honor the commitment, or no longer has any need for the commitment to be honored. More generally, knowledge of potential task relationships can exist for several reasons, either *a priori*, based on the organizational nature of the task structures, based on the history of the negotiations or process by which the task structure was derived, based on the past history of agent interactions, or even through the process of extensive broadcast and discovery. In summary, our goal is to have a suite of coordination mechanisms that can reason about, in a controlled manner, any existing information useful for making coordination, and also acquire in a controlled manner additional information that is important to making the coordination decision in the current situation.

We have also introduced mechanisms in GPGP to coordinate over shared resources. This is accomplished by introducing a new class of commitments called *don't* commitments. They indicate to the local scheduler that a specific task cannot be executed during a certain time interval. Through the use of this type of commitment, a coordination mechanism can introduce serialized access to a resource by a group of agents, or restrict the number of agents that will access the resource during a specific time interval. For example, we have defined a new coordination mechanism that uses a simple negotiation protocol to schedule access to any such mutually exclusive resource [3]. The idea behind this resource-constraint coordination mechanism is that when an agent intends to execute a resource-constrained task, it sends a bid of the time interval it needs and the local priority (projected utility) of its task. After a communication delay, it knows all the bids given out by the other agents at the same time as its own bid. Since all the agents who bid will have the same information about others, if they all use the same commonly accepted rule to decide who will get the time interval, they can get the same result on this round of bidding. The agent who wins will keep its schedule and execute that task within its time interval, and everyone else will mark this time interval with a *don't* commitment, and never try to execute the resource-constrained task in it unless the owner gives it up. All the agents who didn't get their time intervals at the first round will recompute their schedules and bid again.

Contracting in GPGP

In the original conception of GPGP, all coordination decisions were based on agents already intending to perform certain activities as a result of the local goals that they were currently pursuing. There was no way for one agent to get another agent to generate results for it if that agent was not already intending to generate those results. Obviously, for many applications, agents may be structured so that they can pursue a large number of different types of local activities and only when there is a specific need for the result of an activity will the agent want to execute that activity. In order to handle the coordination that arises out of this contracting model of agent interaction, we have introduced a new coordination mechanism with associated additional information in the TAEMS task structure. The development and character of this mechanism is consistent with the work described above on the use of default knowledge and situation-specific control. It is also interesting to note that this new mechanism is valuable even in applications which use bottom-up, data-directed processing strategies such as distributed situation assessment.

The new contracting mechanisms model the *potential* for external interaction using "virtual tasks." A task structure is created at an agent with a "possible-task" or "virtual task" or "partially instantiated" task that abstractly describes results that could be generated by another agent. These results are described in terms of a goal that another agent could satisfy. Virtual tasks are related to

an agent's task structures via *non-local-effect (NLE)* (e.g., enables, facilitates, and so on). If an agent wishes to take advantage of this NLE, it generates a goal, describing its desired outcome, and transmits it to the appropriate agent. The receiving agent then instantiates a task structure, based on the characteristics of the goal. The actual question of whether the receiving agent is capable or interested in performing the task is handled by that agent's problem-solver.

The purpose of the virtual task is to serve as a place-holder in scheduling. An agent can attempt to develop schedules including the contributions of the virtual task to determine whether it can, in fact, develop feasible schedules were that task to be executed. This can take place before the actual communication with the remote agent. It is the modeling of the utility of the virtual task using the stochastic features of the TAEMS framework that distinguishes the GPGP contracting mechanism from a conventional goal-driven system. The incorporation of the TAEMS relationships allows the system to model in advance the contributions of the actions of the remote agent, to reason about the nature and magnitude of the coordination relationships required, and to parameterize the request according to this information, as well as deadlines for transmitting the information represented by the coordination relationships.

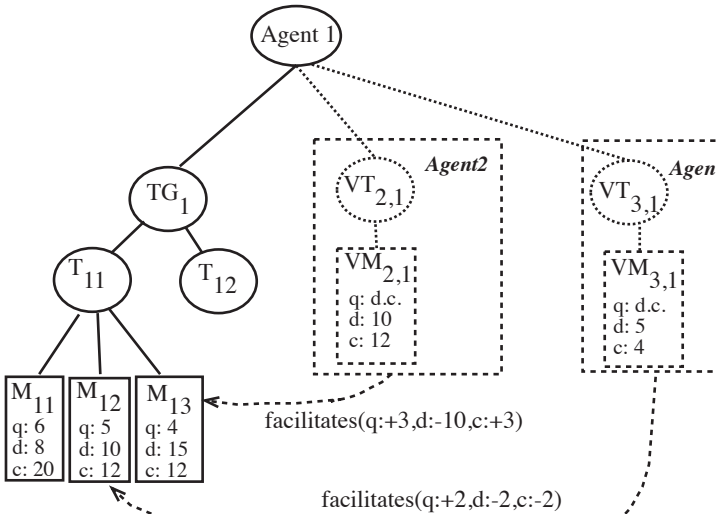

Figure 2: An example of Agent reasoning about how best to contract for needed information from another agent

Consider the idealized situation depicted in Figure 2. Agent 1 has a choice of three methods for achieving task T11. Two of these methods can be facilitated by the actions of other agents. Unlike the original GPGP, there is no presupposition that the tasks will actually be performed unless the agent specifically requests them. Each local method has different performance characteristics. Method 1, for example, has a potential quality of 6, a duration of 8, and a cost of 20. Based purely

on a local perspective, and weighing quality of solution most highly, this would be the method of choice. All other methods produce lower quality solutions, although with lower costs and/or durations. However, by understanding in detail how other agents' activities may interact with its own, the agent is able to predict that, were other agents to execute the appropriate tasks, methods 2 and 3 would be facilitated with quality/cost/duration tuples of (7,8,10) and (7,5,15) respectively. Knowing that higher quality can be achieved, Agent 1 can now reason about the coordination issues involved in ensuring that another agent executes the appropriate task in a timely fashion. First, it can use secondary criteria such as cost and duration to choose which task should be executed by a remote agent. Duration is of consideration both in scheduling later activities and ensuring that sufficient time exists for a request to be transmitted, processed, and for a response to be sent. The agent may or may not have information regarding the cost of executing the tasks by the remote agents and the duration of these tasks— information that might be useful in determining *which* agent to ask for assistance.

Once a remote agent and remote task have been selected, the agent must transmit a request that the task be executed. This request can be annotated with information provided by the local scheduler, indicating the desired parameters under which the task should be executed, including but not limited to the quality, cost, and duration modifiers of the transmitted information, and the preferred finish time of the task.

Use of GPGP with a real problem solver

In using GPGP as part of a real multi-agent system, we have developed a more complex view of the interaction among GPGP/scheduler and the agent's problem-solving component. We have introduced two additional components: a (domain-specific) task assessor and a (domain-specific) task relationship assessor. The problem solver and its associated task assessor use TAEMS to represent the tasks that will likely need to be carried out in the future. The methods that the task assessor provides to the scheduler represent higher level entities, which may still require numerous decisions be made before the actual selection of executable actions is accomplished. It is the problem-solving component that makes these decisions—possibly in a reactive manner. This is one way that we have accommodated the need for some reactivity (without necessitating repeated reschedulings)—by having the task assessor only abstractly plan how to accomplish tasks. The agents' local coordination modules interact, possibly resulting in modifications to the agents' local task structures to represent inter-agent task relationships. This information is used in implementing domain-independent coordination protocols. The coordination-adapted task structures are then used by each agent's real-time scheduler to find the best sequences of activities to meet both the local and network-wide objectives. The resulting schedule of tasks (methods) direct the problem solver,

which ultimately determines what actions to take to best achieve a task. The TAEMS description of the expected tasks created by the task assessor is used to update the current task model that is accessible to the scheduler and coordination module. This triggers the scheduler to generate a new schedule, which is passed back to the problem solver to direct the selection of actions. The schedule consists of an ordered list of methods to be carried out. It implicitly represents the order in which the top-level tasks should be pursued as well as the approaches that should be used to pursue them. The scheduler also associates monitoring information with each method that indicates in what situations the problem solver should alert the scheduler that the partial execution of the method is not performing as expected.

Conclusions and Future Directions

The experiences that we have had in applying GPGP to a number of different applications have led us to believe that our basic approach to coordination, based on quantitative coordination relationships represented in TAEMS 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. Additionally, GPGP can be easily extended to allow the implementation of more top-down and contracting types of coordination mechanisms. It is these extensions that will make GPGP easily integratable into a multi-layered control architecture in which an organizational design defines policies for multi-agent coordination.

Our plans for further extending GPGP fall into two major categories: first, to explore the interplay between local, environment-centered coordination mechanisms and coordination of large agent organizations; and second, to expand the range of possible coordination mechanisms, including those that might make sense only in a large organizational context. With respect to the first goal, our approach is to view the organization as something providing limits, constraints, and other bounds on the local agent coordination process (i.e., making coordination situation-specific). For example, the current GPGP mechanisms are oriented around the discovery and reaction to all possible task relationships. An organizational approach might strictly limit or prespecify the specific relationships that are "important" to coordinate over. Organizational limits might also extend to how various commitments (including pre-established ones) or their associated tasks should be valued. Our current model of commitments, which includes three notions of value (utility, negotiability, and necessity), can easily be expanded in this way. Organizational

information might prespecify certain relationships, or allow for the partial specification of relationships. Finally, the organization might bound the total amount of effort to spend on the meta-task of coordination reasoning and control.

Next, the range of coordination mechanisms that are available must be expanded. We are currently developing additional mechanisms with behaviors such as contracting, sophisticated load-balancing, and coordinating access to mutually exclusive resources, and expect that there will be others that will need to be developed. In our work on distributed data processing, some avoidable errors and inefficiencies have been observed when long sequences of tasks are closely linked across several agents [9]. In such cases the existing mechanisms (developed with the physically linked environment of vehicle monitoring in mind) can cause costly time delays. New mechanisms can apply more complex, multi-stage models of negotiation over commitments. We also want to explore that introduction of mechanisms to support self-interested agents and their interaction with other self-interested agents as well as cooperative agents.

Most importantly, we will also allow the agent designer to introduce specific coordination strategies via a high-level language interface, thus expanding the range of possible coordination strategies. Our approach will be to translate this high-level language into organizational knowledge that will appropriately condition that coordination component (e.g., what coordination relationships to use for particular tasks, what *a priori* commitments need to be initialized with their relative importance, etc.) and allow the agent to be signaled when certain conditions occur in the coordination process (e.g., not being able to meet a commitment). This signaling allows conditionality to be expressed in the high-level language so that in certain situations new organizational knowledge directives can be given to the coordination component. We feel that our 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 ability to dynamically react to emerging situations.

References

1. Keith S. Decker. "Environment Centered Analysis and Design of Coordination Mechanisms." PhD thesis, University of Massachusetts, 1995.

2. Keith S. Decker. "Task environment centered simulation." In M. Prietula, K. Carley, and L. Gasser, editors, *Simulating Organizations: Computational Models of Institutions and Groups*. AAAI Press/MIT Press, 1997.

3. Keith S. Decker and Jinjiang Li. "Coordinated Hospital Patient Scheduling." Submitted to ICMAS-98.

4. 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.

5. Keith S. Decker and Victor R. Lesser. "Designing a family of coordination algorithms." In *Proceedings of the First International Conference on Multi-Agent Systems*, pp. 73–80, San Francisco, June 1995. AAAI Press. Longer version available as UMass CS-TR 94-14.

6. Edmund 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.

7. Nick Jennings. "Commitments and Conventions: The foundation of coordination in multi-agent systems." *The Knowledge Engineering Review*, vol. 8, no. 3, pp. 223–250, 1993.

8. Hector J. Levesque, Philip R. Cohen, and H. T. Nunes. "On acting together." In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 94–99, July 1990.

9. Maram 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,* California, 1997.

10. Tim Oates, M. V. Nagendra Prasad, and V. R. Lesser. "Cooperative Information Gathering: A Distributed Problem-Solving Approach." In *IEE Proceedings on Software Engineering*, Special Issue on Agent-based Systems, Volume 144, No. 1, 1997.

11. Anita Raja, Victor Lesser, and Thomas Wagner. "A more complex view of schedule uncertainty based on contingency analysis." UMass Department of Computer Science Technical Report, December 1997.

12. A.S. Rao and M.P. Georgeff. "BDI agents: From theory to practice." In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 312–319, San Francisco, June 1995. AAAI Press.

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

14. Milind Tambe. "Agent architectures for flexible, practical teamwork." In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Providence, July 1997.

15. Thomas 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.

16. Thomas Wagner, A. Garvey and V. Lesser. "Leveraging Uncertainty in Design-to-Criteria Scheduling." UMass Computer Science Technical Report 1997-11, February 1997.

17. Thomas Wagner, A. Garvey and V. Lesser. "Criteria Directed Task Scheduling." Accepted for publication in *Journal of Approximate Processing* (to appear 1998).

18. Michael Wellman and J. Doyle. "Modular utility representation for decision-theoretic planning." In *Proceedings of the First Internastional. Conference on Artificial Intelligence Planning Systems*, pages 236–242, June 1992.