Investigating Interactions Between Agent Conversations and Agent Control Components*

Thomas Wagner, Brett Benyo, Victor Lesser, and Ping Xuan

Department of Computer Science
University of Massachusetts, Amherst, MA 01003
{wagner,bbenyo,pxuan,lesser}@cs.umass.edu

Abstract. Exploring agent conversation in the context of fine-grained agent coordination research has raised several intellectual questions. The major issues pertain to interactions between different agent conversations, the representations chosen for different classes of conversations, the explicit modeling of interactions between the conversations, and how to address these interactions. This paper is not so ambitious as to attempt to address these questions, only frame them in the context of quantified, scheduling-centric multi-agent coordination research.

1 Introduction

Based on a long history of work in agents and agent control components for building distributed AI and multi-agent systems, we are attempting to frame and address a set of intellectual questions pertaining to agent conversation. *Interaction* lies at the heart of the matter; the issue is interaction between different agent conversations, that possibly occur at different levels of abstraction, but also interaction between the machinery for holding a conversation with other agents and the underlying machinery for controlling the individual agent. Henceforth we will use the term *coordination protocol* to describe the specification for a dialogue between one or more agents that is held for the purpose of coordinating their activities; a *conversation* is an instantiation of a protocol. A *coordination mechanism*, in contrast, denotes a larger grouping of concerns – it is the way in which an agent reasons about interactions, plans to resolve them, and carries out communication activities to do so. We return to the issue of coordination mechanisms in Section 2.3, however, the notion of a mechanism is intertwined in the following intellectual issues:

– Assuming a model where agents are engaged in multiple conversations concurrently, and asynchronously, what are the ramifications of interactions between the different conversations?

* Effort sponsored by the Department of the Navy and Office of the Chief of Naval Research under Grant number N00014-97-1-0591 and by the National Science Foundation under Grant number IIS-9812755. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: 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 Department of the Navy and Office of the Chief of Naval Research, the National Science Foundation, or the U.S. Government.

Should interactions be accounted for at the conversational level or by the underlying agent control components? For example, if an agent is engaged in dialogues with two other agents in an attempt to contract-out two different tasks, e.g., x and y, and the tasks are mutually exclusive, what happens if both tasks are contracted at the same time? Or one is contracted while the other is being negotiated? In our work, recovery would generally take place via decommitment [1], possibly with some penalty involved, but, this response is generally triggered by the agent control components, not the conversation machinery itself.

- Conversations held to coordinate multiple agents generally entail the exchange of task or goal information and temporal constraints. This information may be viewed as particular bindings on variables that are used by the conversation machinery. Using this view, one can envision the conversation machinery querying an oracle (temporal belief-base, truth maintenance system, agent scheduler, etc.) for particular bindings that should be used during the dialogue, e.g., "I can provide you the result by time 10." However, what if multiple candidate tasks are being negotiated that require the same resource(s)?[1] The conversations are clearly interdependent, however, the underlying agent control mechanisms that identify the constrained situation and enumerate possible responses is also part of the interaction. In other words, the involved conversations must query the underlying oracle for information, and in this case, the oracle needs the information from all the conversations in order to make decisions about priorities and what can be accomplished. As soon as one of the conversations results in a committed or intended course of action, the other conversations are impacted. The question is what is the appropriate interface between the conversation machinery and the lower level control components?
- Consider another situation that approaches the same issue from a different perspective. Let α be an agent that has a hard deadline looming and lacks sufficient time to coordinate over all *soft* task interactions (optional coordination points), it must thus modulate the conversation machinery to reflect the upcoming deadline. Options include curtailing conversational activities, i.e., ending existing dialogues or refraining from starting new dialogues, or modifying conversations to reflect the need for haste. The first case involves simply terminating standard dialogues, the second case, however, requires dialogues that are parameterized or include branches that have different temporal requirements (possibly anytime [7, 18, 42] in nature). However, the problem is not that neat – it is actually cyclical. Non-local information obtained via communication influences the agent's beliefs and thus impacts its intentions or planned actions. Thus, continuing a dialogue and gaining more information might actually change the choices that an agent has made and thus result in the agent having more time for conversations. Conversely, time spent conversing may simply detract from domain problem solving. The question is whether or not we must address the issue and if so, what are the implications to the conversational machinery of the agent? Certainly, one can argue that for agents to address real-time and real-resource concerns, the issue must be addressed.

Attempting to frame these questions leads one to consider the implications of agents having multiple, asynchronous, conversations pertaining to different matters and dealing with activities at different levels of abstraction. As discussed in Section 5, intra-level and inter-level interaction in conjunction with interactions between conversations and agent control components pushes harder on the issue of interaction.

These questions are the outcome of an effort to modify our agent coordination technology, namely GPGP [26] and Design-to-Criteria [37, 35], to support openness, situation specificity, and adaptation to different application domains. For example, in a cur-

[1] The issue is more clear if resources are not simple objects that require exclusive access, but are instead sharable, e.g., network bandwidth, where the performance of an action using the resource may degrade based on the state of the resource – and the degrees of degradation vary.

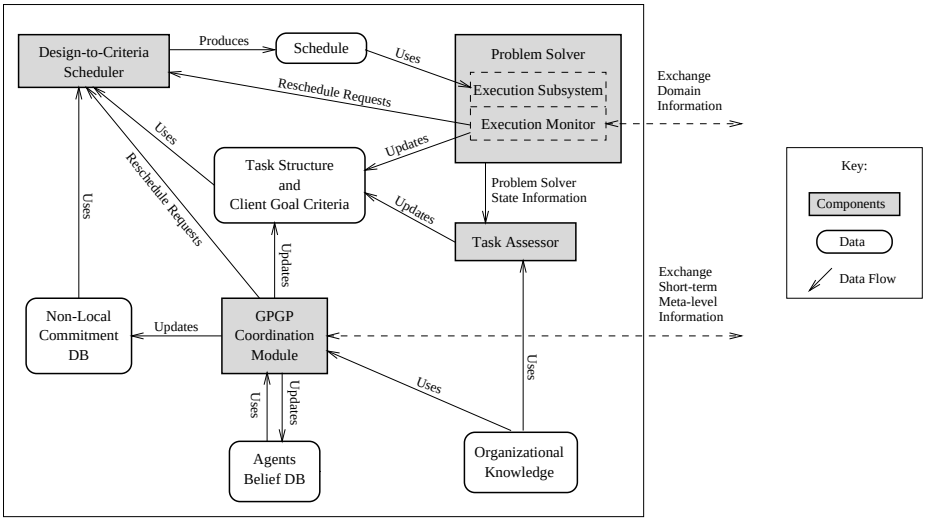

Fig. 1. A Portion of the Prototypical Agent Architecture

rent project we are interfacing our agent control technology with a higher-level process view [21] of the task of sending robot teams into hazardous environments to perform unmanned exploration (e.g., damaged buildings to access structural conditions). This application requires different protocols and different behaviors than applications such as the coordination of agents in an intelligent environment [25], or information gathering agents [8]. In an effort to open GPGP for different applications and to adapt its protocols, we redesigned and reimplemented the important concepts from GPGP and created $GPGP^2$ [40].

It is important to note that while our view of agent control differs from others in the community, from the perspective of the agent conversation, the questions we have posed are relevant to other agent technologies. Perhaps the overall question is the role of agent conversation research and work in multi-agent coordination. On one hand conversational work often focuses on structuring the dialogue between agents [24, 23, 13], or the formal models, motivations, and implications of information exchange [6, 30, 31]. On the other hand, coordination work [29, 33, 34, 19, 14, 9] generally pertains to making decisions about what an agent should do, when, and how it should be done. These two areas of research are related (interdependent?) and we believe both can benefit from cross fertilization and exploring our research ideas, and these conversational issues, in context. Work akin to this has begun using abstractions of the underlying agent machinery or simplified agent task models [13, 30].

Additional context is required to properly frame and understand our questions about interactions and the agent conversational machinery. In some sense, interactions stem from the complexity of the agent control problem. In our work, agents have multiple interacting goals or tasks and multiple different ways to perform them. Agents are also resource bounded and must address real-time and real-resource limitations. The combination of resource limitations and alternative different goals to perform, and alternative different ways to perform them, results in agent control as an optimization style problem

rather than a satisfaction style problem, i.e., the issue becomes evaluation of trade-offs of different alternative courses of action. The interdependencies and the optimization problem view mean that decisions rarely have limited or local scope but instead may impact all of the other choices/decisions made by the agent. In the following sections we clarify by describing our particular view of agent control and our domain independent architecture. We also discuss the finite-state machine approach for coordination protocol specification used in *GPGP2* and return to the questions posed this section.

2 Agent Control Components

We frame the general agent control problem as an action-selection-sequencing activity. Agents have multiple tasks to perform, different ways to perform the tasks, and the control problem is to choose subsets of these for scheduling, coordination with other agents, and execution. The objective of agent control problem solving is to enable agents to meet real-time and real-resource constraints, and to facilitate agent coordination through islands of predictable or stable agent activity.

We approach the control problem from a domain independent perspective, i.e., our research focus is on the construction of generalized agent control components that can be coupled with domain problem solvers, planners, or legacy systems to construct agents suitable for deployment in a multi-agent system. This generalization is achieved by abstracting away from the agents internals. In our work, domain problem solvers describe or translate their problem solving options, their candidate tasks and the primitive actions used to accomplish them, into a task modeling language called TÆMS [11]. The TÆMS models are then passed to generic control components, such as the Design-to-Criteria (DTC) agent scheduler and the (GPGP/*GPGP2*) agent coordination module. Other components include a learning module [32, 20] and a module for system diagnosis [17, 22].

With respect to other approaches to agent control, e.g., BDI-based [28, 4] problem solvers, our tools operate at a different level of detail. We return to this issue in Section 4, though the general idea is that the DTC/GPGP tools perform detailed *feasibility analysis* and *implementation* of high-level goals and tasks selected by other components, like a BDI problem solver. The DTC/GPGP control model assumes that some other component is producing the high-level tasks that the agent is to achieve, either as the result of local-only domain problem solving or as the result of communication (at higher levels) with other agents. A subset of the larger generic agent architecture is shown in Figure 1. In this paper, we describe agent control in the context of the two primary control components, namely the Design-to-Criteria scheduler and the GPGP coordination module.

2.1 TÆMS Task Models

TÆMS (Task Analysis, Environment Modeling, and Simulation) is a domain independent task modeling framework used to describe and reason about complex problem solving processes. TÆMS models are used in multi-agent coordination research [10, 38] and are being used in many other research projects, including: cooperative-information-gathering [27], collaborative distributed design [12], intelligent environments [25], coordination of software process [21], and others [5, 36, 3, 9]. Typically a problem solver

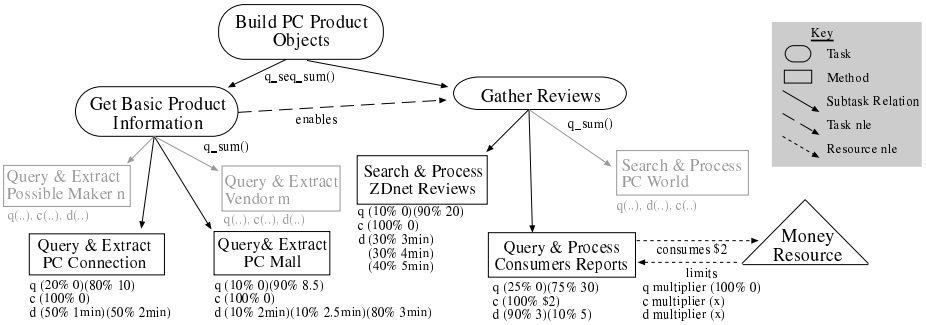

Fig. 2. Simplified Subset of an Information Gathering Task Structure

represents domain problem solving actions in TÆMS, possibly at some level of abstraction, and then passes the TÆMS models on to agent control problem solvers like the multi-agent coordination modules or the Design-to-Criteria scheduler.[2]

TÆMS models are hierarchical abstractions of problem solving processes that describe alternative ways of accomplishing a desired goal; they represent major tasks and major decision points, interactions between tasks, and resource constraints but they do not describe the intimate details of each primitive action. All primitive actions in TÆMS, called *methods*, are statistically characterized via discrete probability distributions in three dimensions: quality, cost and duration. Quality is a deliberately abstract domain-independent concept that describes the contribution of a particular action to overall problem solving. Duration describes the amount of time that the action modeled by the method will take to execute and cost describes the financial or opportunity cost inherent in performing the action. Uncertainty in each of these dimensions is implicit in the performance characterization – thus agents can reason about the certainty of particular actions as well as their quality, cost, and duration trade-offs. The uncertainty representation is also applied to task interactions like enablement, facilitation and hindering effects, [3] e.g., "10% of the time facilitation will increase the quality by 5% and 90% of the time it will increase the quality by 8%." The quantification of methods and interactions in TÆMS is not regarded as a perfect science. Task structure programmers or problem solver generators *estimate* the performance characteristics of primitive actions. These estimates can be refined over time through learning and reasoners typically replan and reschedule when unexpected events occur.

To illustrate, consider Figure 2, which is a conceptual, simplified sub-graph of a task structure emitted by the BIG [27] information gathering agent; it describes a portion of the information gathering process. The top-level task is to construct product models of retail PC systems. It has two subtasks, *Get-Basic* and *Gather-Reviews*, both of which are decomposed into methods, that are described in terms of their expected quality, cost,

[2] In the process work, a translator transforms and abstracts process programs into TÆMS task structures for scheduling and coordination.

[3] Facilitation and hindering task interactions model soft relationships in which a result produced by some task may be beneficial or harmful to another task. In the case of facilitation, the existence of the result, and the activation of the nle generally increases the quality of the recipient task or reduces its cost or duration.

and duration. The *enables* arc between *Get-Basic* and *Gather* is a non-local-effect (nle) or task interaction; it models the fact that the review gathering methods need the names of products in order to gather reviews for them. Other task interactions modeled in TÆMS include: *facilitation*, *hindering*, *bounded facilitation*, *sigmoid*, and *disablement*. Task interactions are of particular interest to coordination research because they identify instances in which tasks assigned to different agents are interdependent – they model, in effect, implicit joint goals or joint problem solving activity. Coordination is motivated by the existence of these interactions.

Returning to the example, *Get-Basic* has two methods, joined under the *sum()* quality-accumulation-function (*qaf*), which defines how performing the subtasks relate to performing the parent task. In this case, either method or both may be employed to achieve *Get-Basic*. The same is true for *Gather-Reviews*. The qaf for *Build-PC-Product-Objects* is a *seq_sum()* which indicates that the two subtasks must be performed, in order, and that their resultant qualities are summed to determine the quality of the parent task; thus there are nine alternative ways to achieve the top-level goal in this particular substructure. In general, a TÆMS task structure represents a family of plans, rather than a single plan, where the different paths through the network exhibit different statistical characteristics or trade-offs.

TÆMS also supports modeling of tasks that arrive at particular points in time, individual deadlines on tasks, earliest start times for tasks, and non-local tasks (those belonging to other agents). In the development of TÆMS there has been a constant tension between representational power and the combinatorics inherent in working with the structure. The result is a model that is non-trivial to process, coordinate, and schedule in any optimal sense (in the general case), but also one that lends itself to flexible and approximate processing strategies.

2.2 Design-to-Criteria Scheduling: Local Agent Control

The Design-to-Criteria (DTC) scheduler is the agent's local expert on making control decisions. The scheduler's role is to consider the possible domain actions enumerated by the domain problem solver and choose a course of action that best addresses: 1) the local agent's goal criteria (its preferences for certain types of solutions), 2) the local agent's resource constraints and environmental circumstances, and 3) the non-local considerations expressed by the GPGP coordination module. The general idea is to evaluate the options in light of constraints and preferences from many different sources and to find a way to achieve the selected tasks that best addresses all of these.

The scheduler's problem is framed in terms of a TÆMS task structure emitted by the domain problem solver. Scheduling problem solving activities modeled in the TÆMS language has four major requirements: 1) to find a set of actions to achieve the high-level task, 2) to sequence the actions, 3) to find and sequence the actions in soft real-time, 4) to produce a schedule that meets dynamic goal criteria, i.e., cost, quality, duration, and certainty requirements, of different clients. TÆMS models multiple approaches for achieving tasks along with the quality, cost, and duration characteristics of the primitive actions, specifically to enable TÆMS clients to reason about the trade-offs of different courses of action. In other words, for a given TÆMS task model, there are multiple approaches for achieving the high-level task and each approach has different quality, cost, duration, and certainty characteristics. In contrast to classic scheduling problems,

the TÆMS scheduling objective is not to sequence a set of unordered actions but to *find and sequence* a set of actions that *best* suits a particular client's quality, cost, duration, and certainty needs. Design-to-Criteria is about examining the current situation, the current options before the agent, and deciding on a course of action – it is about targetable contextual decision making.

Design-to-Criteria scheduling requires a sophisticated heuristic approach because of the scheduling task's inherent computational complexity ($\omega(2^n)$ and $o(n^n)$) it is not possible to use exhaustive search techniques for finding optimal schedules. Furthermore, the deadline and resource constraints on tasks, plus the existence of complex task interrelationships, prevent the use of a single heuristic for producing optimal or even "good" schedules. Design-to-Criteria copes with these explosive combinatorics through approximation, criteria-directed focusing (goal-directed problem solving), heuristic decision making, and heuristic error correction. The algorithm and techniques are documented more fully in [37, 35].

2.3 GPGP Coordination: Managing Non-Local Interactions

GPGP (Generalized Partial Global Planning) is the agent's tool for interacting with other agents and coordinating joint activity. GPGP is a modularized, domain independent, approach to scheduling-centric coordination. In GPGP, coordination *modulates* local control by posting constraints on an agent's local DTC scheduler. The GPGP coordination module is responsible generating communication actions, that is communicating with other agents (via their local communication modules), and making and breaking task related *commitments* with other agents. The coordination module is comprised of several modular coordination mechanisms, subsets of which may be applied during coordination depending on the degree of coordination desired. More specifically, GPGP defines the following coordination mechanisms (for the formal details see [10]):

1. **Share Non-Local Views** - This most basic coordination mechanism handles the exchange of local views between agents and the detection of task interactions. Exchanging local views is the only way in which agents can detect and coordinate over task interactions. The mechanism exchanges information, or not, according to three different exchange policies: *exchange none*, where no information is exchanged; *exchange some*, where only part of the local view is communicated; and *exchange all*, where the entire local view is communicated. This coordination mechanism is necessary for all other coordination mechanisms – without a local view of non-local tasks and an understanding of existing task interactions there is nothing to coordinate.
2. **Communicate Results** - This coordination mechanism handles communicating the results of method execution to other agents. It is governed by three different policies: the *minimal policy* where only the results necessary to satisfy external commitments are communicated; the *task-group policy* where all the minimal results plus the final results for a task group are communicated; and the *all* policy where all results are communicated. This mechanism is meaningless without mechanism 1 above or the following mechanisms that form commitments.
3. **Avoid Redundancy** - This mechanism deals with detected redundancy by picking an agent at random to execute the redundant method in question. The agent then becomes committed to performing the action and the other agents will have non-local commitments denoting that some other agent will carry out the task at a predetermined time. Note, the type of redundancy in question here is simple duplication of work, in contrast to the redundancy of being able to generate a similar result using different methods.

4. **Handle Hard Task Relationships** - The *enables* NLE pictured in Figure 2 denotes a hard task relationship. This coordination mechanism deals with such hard, non-optional, task interactions by committing the predecessors of the *enables* to perform the task by a certain deadline.
5. **Handle Soft Task Relationships** - Soft task interactions, unlike hard interactions like *enables*, are optional. When employed, this coordination mechanism attempts to form commitments on the predecessors of the soft interactions to perform the methods in question before the methods that are on the receiving end of the interaction.

As mentioned above, the GPGP coordination module modulates local control by placing constraints, called *commitments*, on the local scheduler. The commitments represent either deals that GPGP has made with other agents, e.g., agreeing to perform method M by time T, or deals that GPGP is considering making with other agents. The commitments fall into four categories:

Deadline Commitment This type of commitment denotes an agreement to execute a particular method by a particular time. Thus if agent A needs the results from a method execution being performed by another agent, agent B, and they form a deadline commitment, agent A can then plan other activities based on the expectation of receiving the results from B by the deadline T.

Earliest Start Time Commitment This commitment denotes an agreement not to start executing a particular method prior to an agreed upon time. This type of commitment is the converse of the deadline commitment. In the two agent scenario above, this commitment could be used to denote that while agent B should execute M by time T, it should also not start executing M before time T'.

Do Commitment This commitment is weak and simply denotes a commitment to execute a particular method at some time.

Don't Commitment This commitment denotes an agreement not to perform a particular method during a particular interval. It is particularly useful for coordination over shared resources.

Salient features of GPGP-based coordination include a domain independent approach to coordination, exchange of non-local information to construct a partial global view, a *worth* driven view of tasks and actions (from TÆMS), different information exchange policies for many of the coordination mechanisms, a subset of mechanisms that are independent and can be applied, or not, depending on the current context (e.g., looming deadlines).

Figure 3 shows a multi-agent problem solving situation in which an information gathering task structure (akin to Figure 2) is distributed across several agents. The high-level objective is to build product objects. The two subtasks are to build objects for PC software products, and to build objects for Mac products. Note that the actions used to perform tasks like *Gather-Reviews* are abstracted out of this figure. The entire PC related branch of the tree is contracted out to a single agent, *Task Agent A*, while the Mac related branch is broken down and contracted out to two other agents, *Task Agents B* and *C*. There are interactions between the *Get-Basic-Product-Information* tasks and the *Gather-Reviews* tasks, as well as interactions between the PC and Mac versions of these tasks (products may be multi-platform). Using GPGP, the agents coordinate as follows:

– Step 1: Exchange local views. Agents A, B, and C exchange their local views, i.e., they exchange portions of their task structures. This gives each agent a limited view of the activities being performed by the other agents.

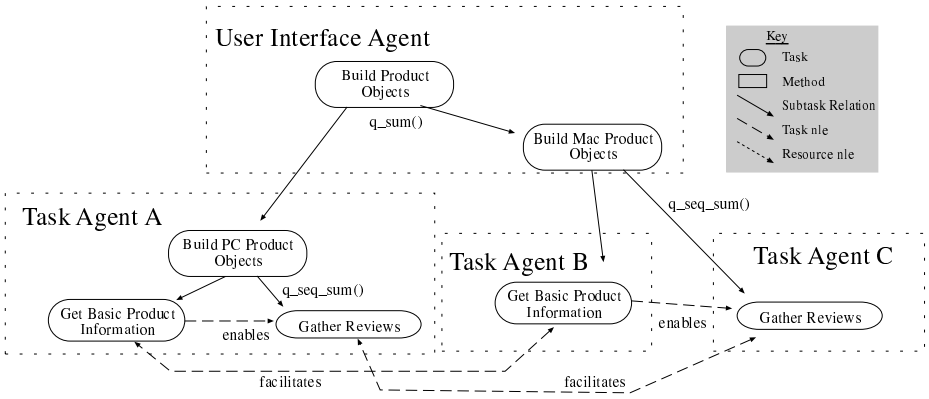

Fig. 3. Multiple Interacting Information Agents

- Step 2: Detect interactions. In this case, the interactions may be specified *a priori* by the *User Interface Agent*. However, if the interface agent did not have a complete view of the task beforehand, the agents will compare inputs and outputs of their different actions and match up relationships accordingly.
- Step 3: Coordinate over interactions. Agent A has mutual facilitations with agents B and C. Agent B has a mutual facilitation with agent A, as well as an enables relationship with C. C has a mutual facilitation with A, but also requires input from B in order to do its problem solving. The sequencing and interaction of coordination over these interactions is one of the issues of this paper, however, in general, the interactions are handled by:
  1. Agent B evaluating its intended course of action and offering agent C a deadline commitment that specifies the deadline by which it will produce a result so that agent C can execute.
  2. Agent A evaluating its intended course of action and offering a commitment to agent B that specifies when a portion of the results for A's *Get-Basic-Product-Information* will be available.
  3. Agent B evaluating its schedule and offering agent A a similar commitment about the partial results of its *Get-Basic-Product-Information* task.
  4. Agent A, after considering its schedule, will then offer agent C a commitment about when the partial results of its *Gather-Reviews* task will be available.
  5. Agent C will offer a similar commitment to agent A about its *Gather-Reviews* task's results.
- Step 4: Execute, recommit, and exchange. The agents will then perform their scheduled primitive actions, rescheduling and recommitting if necessary, and exchanging results as specified by the commitments they have formed.

As mentioned, coordination or agent conversation must rely on an underlying oracle or analysis procedures to determine bindings on particular variables that are exchanged during the agent dialogue. For example, an agent must have a good idea of when a particular result can be provided to another agent in order to propose a commitment to that effect. In the GPGP/DTC world view, this information is generally provided by the scheduler. However, GPGP also requires non-scheduler analysis code, for example, code to detect task interactions or to determine which information policy should be used. Thus, GPGP mechanisms embody both analysis aspects of the coordination

problem and coordination protocol aspects. The problem is that this integration of concerns makes extending the protocols difficult – they are, in essence, built into the code and isolated from the outside world. *GPGP2* addresses this problem by separating the analysis procedures from the specification of the agent coordination protocol.

3 *GPGP2*

The *GPGP2* label on our current generation of agent coordination tools is primarily for historical reasons. The goal of the *GPGP2* project is to develop a new approach to specifying coordination *mechanisms* that separates the coordination protocol from the supporting analysis code so that coordination protocols may be easily modified and adapted for particular contexts. One step in the verification of the new tools is to reimplement the functionality of GPGP, including its fairly simple coordination protocols and one-shot coordination nature. However, the main objective is to take the work beyond the territory already covered by GPGP.

Whereas GPGP grouped analysis functionality and protocol specification into a single body of embedded code, *GPGP2* takes a very different approach. Coordination protocols are specified using an extended finite state machine (FSM) model where states denote conversational states and transitions are associated with calls to communication actions or analysis code. This approach to specification is widespread and akin to AgenTalk [23] and COOL [2], but the work differs in the way in which conversations interact with the underlying agent control machinery. Implementationally, FSMs are specified via scripts that are processed by a java-based FSM interpreter. The interpreter emits java code that is then incorporated into a *coordination bean* which is integrated into the generic java agent framework [16]. The coordination bean interacts with the rest of the agent components through an event/registration mechanism and by direct invocation when using certain support features of the framework. Features of the FSM model / interpreter include:

- Support for multiple concurrent asynchronous conversations between a given agent and other agents in the environment.
- FSM variables enabling protocols to store information explicitly – in addition to the implicit information contained in each conversation state. For example, to store the commitment time last proposed by another agent.
- Shared FSM variables that enable different conversations (FSM instances) to interact. For example, conversations focused on a particular set of interrelated tasks (possibly sequentially dependent) might contain points of synchronization to serialize their efforts. The synchronization phase would entail a shared semaphore-like variable and the passing of particular bindings. This information could also be passed outside of the coordination bean via the standard agent data structures / knowledge bases, but, intuitively it seems more efficient to do this sort of operation inside the coordination machinery rather than through the general agent control structures. This is a design decision, but, it is the embodiment of the issue of handling interactions between different conversations. It is unclear, at this time, which is the right approach and unclear as to whether or not a stronger, explicit, representation of conversation interaction is needed.
- Timers enable machines to set timers and then block, waiting for particular events to occur. The timers enable conversations to time-out if responses are not produced within a given window. The timeout duration can be specific to the conversation or a global default used by all conversations.

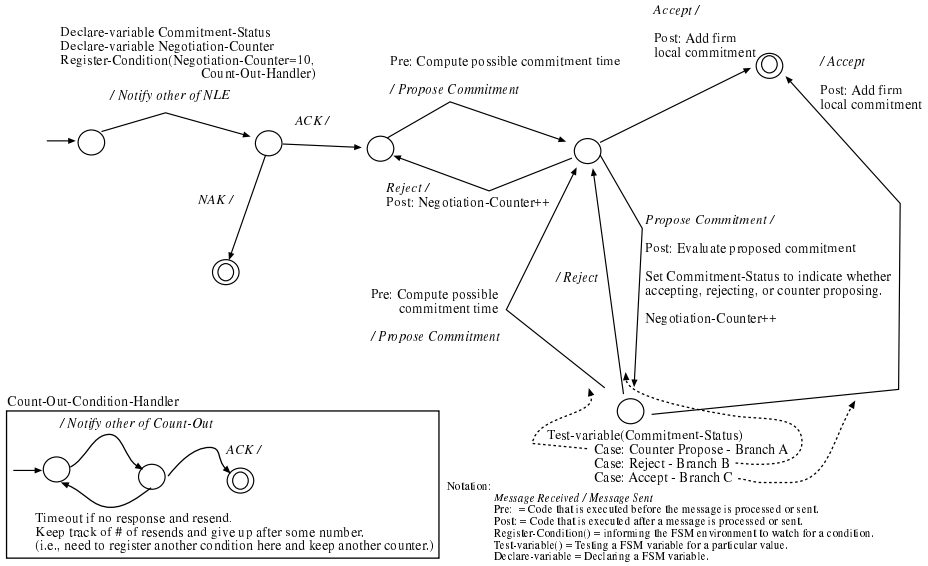

Fig. 4. Initiator FSM to Coordinate Hard Task Interaction

- Event registration and creation. Events may be generated from within the FSMs as well as from within the agent. In effect, each conversation is a first class object within the agent framework in terms of event generation and event catching. Conversations can thus interact even without explicit *a priori* knowledge of which other conversations are likely to interact.
- As part of the event mechanism, FSMs can initiate other conversations, i.e., one FSM may detect the need for a new dialogue and can fire-up a new FSM to handle the dialogue.
- Inheritance. Coordination protocols can be subclassed and specialized. This facilitates rapid protocol development and simple specialization of existing protocols.
- Pre and post conditions on transitions. Transitions may have a set of actions (including tests of FSM variables) that take place before the transition and sets of actions that take place as the transition completes.
- Exceptions. FSMs may throw and catch exceptions. This allows FSMs to handle timeout conditions and other events that drastically change the execution flow through the FSM, and to do so in a succinct fashion. The alternative is complete specification of all exceptions as transitions from each state.

Figure 4 show an example of an initiator FSM[4] to handle the coordination of a hard task interaction (the temporal sequencing of task performance). The FSMs in the figure are designed to handle the formation of a single commitment. One of the outstanding research questions is determining the appropriate grainsize for an agent conversation. We are currently using a model where conversations and task interaction coordination are 1:1. However, consider a case where there is an interaction from task α to β, and then from β to γ. Chains of such interactions may require one conversation to coordinate the chain of interactions, rather than multiple independent conversations or multiple

[4] Generally the responder is a reflected version of the initiator, in these cases, it is probably reasonable to specify a single FSM and then adapt the interpreter to output two versions. This would remove the need to analyze the FSMs for reachability and related issues.

conversations that interact via shared variables. Relatedly, consider a case where agent A and agent B have multiple different task interactions. With our current model, these will be handled by multiple concurrent and asynchronous conversations between the agents. However, they could also be handled by a single conversation that dealt with the multiple task interactions at once. In both cases, interactions between the FSMs are at issue. In the first case, the conversations are interdependent because the tasks over which they are coordinating are interdependent. In the second case, the conversations are interdependent because the tasks are associated with the same agents, i.e., the interdependence is not between the tasks per se, but, stems from the particular assignment of tasks to agents.

4 Interactions Revisited

The issue of interactions is potentially larger than described in Section 1. We have thus far identified the issue of interactions between different conversations, and interactions between the conversation machinery and the agent control machinery. However, we are currently considering new agent dialogues or coordination mechanisms that potentially operate at a higher-level than the conversations held to perform GPGP style coordination.

GPGP and *GPGP2* deal with the temporal sequencing of tasks and with exploring different tasks and constraints assigned to a set of agents. In some sense, this style of coordination is about feasibility analysis and solution enactment based on the assumption that tasks are generated dynamically during problem solving by the agent problem solver or by an external (possibly human) client. In other words, GPGP assumes that some other process is responsible for determining the set of candidate tasks to schedule and coordinate. Note that TÆMS models alternative different ways to perform tasks, and does so hierarchically, so the GPGP problem is not simply to coordinate and schedule a set of primitive actions that must be performed but instead to *choose* which actions to perform based on utility and feasibility. However, GPGP's (and DTC's) choices are limited to the set of tasks and actions emitted by the problem solver. All GPGP conversations pertain to the detection of interactions, the sequencing of activities to resolve interactions, and the sharing of results; they do not pertain to the determination of the high-level goals of the agent.

Our current work in integrating *GPGP2* with a process-level controller, however, requires that we address the issue of task allocation to agents and the determination of which tasks to perform from a more global perspective. Note that these are two separate, but similar, issues. Task allocation is the problem of assigning members of a set of tasks, say $\mathcal{T}$, to individual agents belonging to a set of candidate agents. This requires knowledge about the capabilities and resources of the agents and knowledge about the structure of the tasks (possibly a high-level view of interdependence or ordering). The determination of which tasks the overall agent network should pursue is a different matter – this is the process of generating $\mathcal{T}$. Both of these activities require that agents be able to engage in conversations other than those required for GPGP-style coordination. These conversations must convey information such as the capabilities of the agents but also information pertaining to the state of the overall problem solving network. It appears that these conversations pertain to different concerns and operate at different

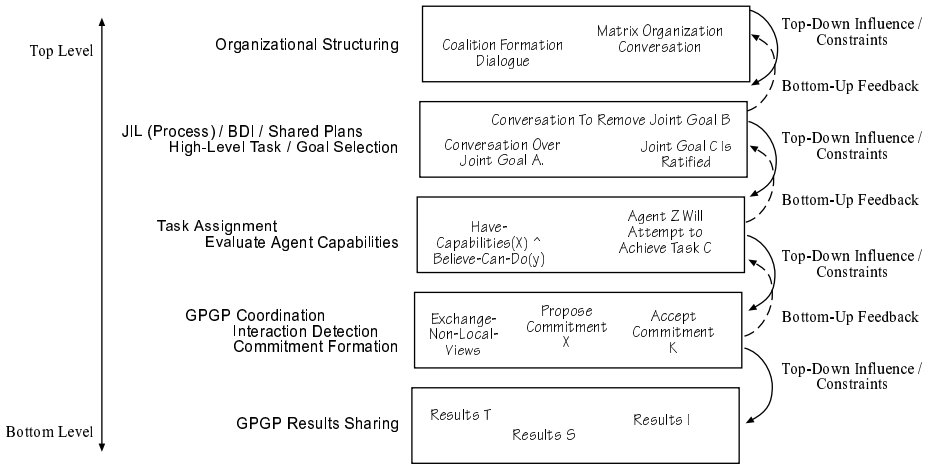

Fig. 5. Conversational Levels and Interactions

levels of detail.[5] However, there is clearly an interaction between the production of $\mathcal{T}$, the assignment of members of $\mathcal{T}$ to agents, and the feasibility of the tasks, i.e., in this case we are faced with interactions between the conversations held to determine overall objectives, conversations held to determine task assignment, and conversations held to determine task feasibility and task performance. Additionally, these conversations are asynchronous; not just with respect to the different levels, but there might be different conversations at each level going on simultaneously. Figure 5 illustrates this idea. In some sense, decisions made at the upper levels set the policy for conversations at the lower levels. For example, deciding to pursue tasks α and β at the upper level determine that at the GPGP-level, conversations will be held toward that ends. However, there is also a feedback process in which the lower-level must explore the feasibility of the tasks selected by the upper levels. Consider a situation in which a set of tasks are selected but when the agents attempt to coordinate, sequence, and perform the required actions it is discovered that the agent network lacks sufficient resources to carry out the activities (recall, we address problems where task interactions and temporal constraints make it difficult to ascertain what is possible without actually going through the process of attempting to coordinate and schedule the activities). In this case, the choice of which tasks to pursue for the overall network must be modified.[6] Again, we return to the issue of interaction. Should these interactions be explicitly modeled and handled by the

[5] We are currently also exploring the integration of our temporal/constraint based coordination with BDI approaches to agent control. We believe that a BDI framework can be used in the upper level of the agent control to determine which tasks to perform from a coarse-grained perspective (intentions). The fine-grained coordination and scheduling of the activities is then carried out by our tools.

[6] An alternative is to provide the lower-level feasibility and implementation tools with a larger view of the space of candidate tasks. In this model, the lower-level tools could provide guidance about which tasks should be pursued at the higher-levels based on the analysis. Note that in this case, the upper and lower-levels have essentially the same information, just at different levels of abstraction.

conversation machinery? Does this require a negotiation style interface [15] between the different conversational levels? Relatedly, should there be different conversational machinery for these different levels of conversation?

Once one begins regarding agent conversation as being stratified, other levels become obvious. Work in organizing the computation and organizing multi-agent systems obviously entails conversations that take place at yet another (higher) level of abstraction. In these conversations agents determine the structure in which the problem solving will take place. Again, conversations at this level appear to interact with the lower levels, and vice versa. Again, are new representations needed? Is new machinery needed to hold conversations of this type? We have recently developed new modeling constructs and reasoning tools for addressing local agent control at these higher levels [39, 35], but the question of interaction and machinery remains.

The stratification also moves down the food chain. If we examine GPGP, there are clearly two different levels of conversation within GPGP itself. At one level, agents exchange local information to construct partial global views of the rest of the world. The agents then carry out dialogues to attempt to handle various task interactions. These activities fall under the general umbrella of feasibility and solution enactment. However, the act of communicating results can be viewed as a different type of activity. In *GPGP2*, the same machinery is used to communicate results as to carry out the other activities, but, the activities are inherently different. In this case it appears that new representations and machinery are not needed, possibly because the interactions between these levels are one way – results being communicated does not affect existing conversations, though the results may cause agents to engage in new conversations with other agents as their problem solving state evolves.

5 Conclusion

We have attempted to identify the issue of interactions in agent conversations and to provide the reasons that interactions are a research question worth addressing. In summary, we believe that both the agent conversation community and the coordination community could benefit from the integration of our technologies and that the meaningful integration of these technologies leads to the issue of interaction between the conversational level and the control level. Additionally, based on our work in coordination, we hypothesize that different levels of interacting, asynchronous, conversations are necessary to scale multi-agent systems for deployment in complex, open environments. The main issues are what representations or formalisms are useful and whether or not explicitly representing and reasoning about interactions is required.

Stepping aside from the notion of levels and interactions – there is also the issue of uncertainty in conversations and uncertainty in agent coordination. In TÆMS we explicitly represent, and reason about, the certainty of actions. We have begun to reason about the role of uncertainty in GPGP-style coordination [41], but, it seems intuitive that the uncertainty question is ubiquitous and applies to all levels of agent conversation.

References

1. M. Andersson and T. Sandholm. Leveled commitment contracts with myopic and strategic agents. In *Proc. of the 15th Nat. Conf. on AI*, pages 38–44, 1998.

2. M. Barbuceanu and M.S. Fox. COOL: A language for describing coordination in multi agent syste ms. In *Proc. of the First Intl. Conf. on Multi-Agent Systems (ICMAS95)*, pages 17–25, 1995.
3. A.L.C. Bazzan, V. Lesser, and P. Xuan. Adapting an Organization Design through Domain-Independent Diagnosis. UMASS CS Technical Report TR-98-014, February 1998.
4. M.E. Bratman. *Intention Plans and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.
5. N. Carver and V. Lesser. The DRESUN testbed for research in FA/C distributed situation assessment: Extensions to the model of external evidence. In *Proc. of the First Intl. Conf. on Multiagent Systems*, June, 1995.
6. P.R. Cohen and H.J. Levesque. Communicative actions for artificial agents. In *Proc. of the First Intl. Conf. on Multi-Agent Systems (ICMAS95)*, page 445, 1995.
7. T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proc. of the Seventh Nat. Conf. on Artificial Intelligence*, pages 49–54, St. Paul, Minnesota, August 1988.
8. K. Decker, M. Williamson, and K. Sycara. Intelligent adaptive information agents. *Journal of Intelligent Information Systems*, 9:239–260, 1997.
9. K. Decker and J. Li. Coordinated hospital patient scheduling. In *Proc. of the Third Intl. Conf. on Multi-Agent Systems (ICMAS98)*, pages 104–111, 1998.
10. K. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms*. PhD thesis, University of Massachusetts, 1995.
11. K. 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, 1996.
12. K. Decker and V. Lesser. Coordination assistance for mixed human and computational agent systems. In *Proc. of Concurrent Engineering 95*, pages 337–348, McLean, VA, 1995. Concurrent Technologies Corp. Also available as UMASS CS TR-95-31.
13. M. d'Inverno, D. Kinny, and M. Luck. Interaction protocols in agentis. In *Proc. of the Third Intl. Conf. on Multi-Agent Systems (ICMAS98)*, pages 261–268, 1998.
14. E.H. Durfee and T.A. Montgomery. Coordination as distributed search in a hierarchical behavior space. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1363–1378, 1991.
15. A. Garvey, K. Decker, and V. Lesser. A negotiation-based interface between a real-time scheduler and a decision-maker. In *AAAI Workshop on Models of Conflict Management*, Seattle, 1994. Also UMASS CS TR–94–08.
16. B. Horling. A Reusable Component Architecture for Agent Construction. UMASS CS Technical Report TR-1998-45, October 1998.
17. B. Horling, V. Lesser, R. Vincent, A. Bazzan, and P. Xuan. Diagnosis as an Integral Part of Multi-Agent Adaptability. UMASS CS Technical Report TR-99-03, January 1999.
18. E.J. Horvitz. Reasoning under varying and uncertain resource constraints. In *Proc. of the Seventh Nat. Conf. on AI*, August 1988.
19. N.R. Jennings. Controlling cooperative problem solving in industrial multi-agent systems. *AI*, 75(2):195–240, 1995.
20. D. Jensen, M. Atighetchi, R. Vincent, and V. Lesser. Learning Quantitative Knowledge for Multiagent Coordination. *Proc. of AAAI-99*, 1999. Also as UMASS CS Technical Report TR-99-04.
21. S.M. Sutton Jr. and L.J. Osterweil. The design of a next-generation process language. In *Proc. of the Fifth ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 142–158, September 1997.
22. G.A. Kaminka and M. Tambe. What is Wrong With Us? Improving Robustness Through Social Diagnosis. In *Proc. of the 15th Nat. Conf. on AI*, July 1998.

23. K. Kuwabara, T. Ishida, and N. Osato. Agentalk: Coordination protocol description for multi-agent systems. In *Proc. of the First Intl. Conf. on Multi-Agent Systems (ICMAS95)*, page 445, 1995.
24. Y. Labrou and T. Finin. A Proposal for a new KQML Specification. Computer Science Technical Report TRCS-97-03, University of Maryland Baltimore County, February 1997.
25. V. Lesser, M. Atighetchi, B. Horling, B. Benyo, A. Raja, Regis Vincent, Thomas Wagner, Ping Xuan, and Shelley XQ. Zhang. A Multi-Agent System for Intelligent Environment Control. In *Proc. of the Third Intl. Conf. on Autonomous Agents (Agents99)*, 1999.
26. V. Lesser, K. Decker, N. Carver, A. Garvey, D. Neiman, N. Prasad, and T. Wagner. Evolution of the GPGP Domain-Independent Coordination Framework. UMASS CS Technical Report TR-98-05, January 1998.
27. V. Lesser, B. Horling, F. Klassner, A. Raja, T. Wagner, and S. XQ. Zhang. BIG: A resource-bounded information gathering agent. In *Proc. of the 15th Nat. Conf. on AI (AAAI-98)*, July 1998.
28. A.S. Rao and M.P. Georgeff. Modelling rational agents within a BDI-architecture. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proc. of the 3rd Intl. Conf. on Principles of Knowledge Representation and Reasoning*, pages 473–484. Morgan Kaufmann, 1991.
29. S. Sen and A. Biswas. Effects of misconception on reciprocative agents. In *Proc. of the 2nd Intl. Conf. on Autonomous Agents (Agents98)*, pages 430–435, 1998.
30. M. P. Singh. Developing formal specifications to coordinate heterogenous autonomous agents. In *Proc. of the Third Intl. Conf. on Multi-Agent Systems (ICMAS98)*, pages 261–268, 1998.
31. I. Smith, P.R. Cohen, J.M. Bradshaw, M. Greaves, and H. Holmback. Designing conversation policies using joint intention theory. In *Proc. of the Third Intl. Conf. on Multi-Agent Systems (ICMAS98)*, 1998.
32. T. Sugawara and V.R. Lesser. Learning to improve coordinated actions in cooperative distributed problem-solving environments. *Machine Learning*, 1998.
33. M. Tambe, J. Adibi, Y. Al-Onaizan, A. Erdem, G. Kaminka, S. Marsella, I. Muslea, and M. Tallis. ISIS: Using an Explicit Model of Teamwork in RoboCup'97. In *Proc. of the First Robot World Cup Competition and Conf.*, 1997.
34. M. Tambe. Towards flexible teamwork. *Journal of AI Research*, 7:83–124, 1997.
35. T. Wagner. *Toward Quantified Control for Organizationally Situated Agents*. PhD thesis, University of Massachusetts at Amherst, February 2000.
36. R. Vincent, B. Horling, T. Wagner, and V. Lesser. Survivability simulator for multi-agent adaptive coordination. In *Proc. of the First Intl. Conf. on Web-Based Modeling and Simulation*, 1998.
37. T. Wagner, A. Garvey, and V. Lesser. Criteria-Directed Heuristic Task Scheduling. *Intl. Journal of Approximate Reasoning, Special Issue on Scheduling*, 19(1-2):91–118, 1998. A version also available as UMASS CS TR-97-59.
38. T. Wagner and V. Lesser. Toward Ubiquitous Satisficing Agent Control. In *1998 AAAI Symposium on Satisficing Models*, March, 1998.
39. T. Wagner and V. Lesser. Relating quantified motivations for organizationally situated agents. In N.R. Jennings and Y. Lespérance, editors, *Intelligent Agents VI (ATAL-99)*, Lecture Notes in AI. Springer-Verlag, Berlin, 2000.
40. T. Wagner, V. Lesser, B. Benyo, A. Raja, P. Xuan, and S. XQ Zhang. GPGP2: Improvement Through Divide and Conquer. Working document, 1998.
41. P. Xuan and V. Lesser. Incorporating uncertainty in agent commitments. In N.R. Jennings and Y. Lespérance, editors, *Intelligent Agents VI (ATAL-99)*, Lecture Notes in AI. Springer-Verlag, Berlin, 2000.
42. S. Zilberstein and S. Russell. Efficient resource-bounded reasoning in AT-RALPH. In *Proc. of the First Intl. Conf. on AI Planning Systems*, College Park, Maryland, June 1992.