Negotiation and Coordination (extended abstract)

Keith Decker
Dept. of Computer and Information Sciences
University of Delaware
Newark, DE 19711
decker@cis.udel.edu

October 5, 1998

Using the definitions previously given by workshop participants as a starting point, we can construct a working definition of negotiation (as some kind of process) by looking at three things: constraints on the entities involved in the process; contraints on the process itself; constraints on the outcomes of the process. All three components are important, as a system not meeting constraints in all three areas will not be considered as a negotiating system.

Constraints on participating entities: Let's assume that "arranging (ordering) some activities" is a suitable negotiation process outcome (I'll discuss constraints on outcomes in a bit). Most people would agree that more than one entity is required for a negotiation process (that is, when a person arranges thier own activities alone, we wouldn't call that process a "negotiation"). The other constraint is on the autonomy of the entity with respect to the negotiation process outcome. Whether the entities involved are subroutines, software agents, smart hardware devices, animals, humans, or multi-national corporations, then if the outcome concerns (for example) an arrangement of activities, then the entities must have the ability to effect the outcome (in this case, alter activitiy arrangement), even if the effect is indirect.

Constraints on the negotiation process itself: It would appear that there are also few constraints on the negotiation process itself. As Sandip Sen pointed out, there is no need for direct communication (one or several mediators may be used). However, archetypical stages are often observed (e.g. propose / accept / reject / counterpropose). This looseness is to be expected because the object of a negotiation is often self-referentially the negotiation process itself. Unfortunately none of these stages can be considered strictly necessary since processes can be constructed so as to make one or more of these stages tacit or implicit. On the other hand, the majority of recognizable negotiation processes do have such observable stages (from couples negotiating over what movie to see; through sales,contract, or labor negotiation; to governmental bodies negotiating over laws and treaties). Proposals are sometimes elided because the agents share a physical environment that embodies the proposals directly (as sugested by Van Parunak). Explicit acceptance or rejection can be replaced by implicit execution of key actions.

It should also be mentioned that sometimes there are strong contraints imposed on a particular negotiation process (usually, external constraints) that can lead to situations where the process takes on more importance than the potential outcomes. Indeed, the principals may be well aware that no altered outcome is actually expected. By this I mean the kind of classic, ingenuous "institutional" response such as observed by Meyer and Rowan [23] with respect to government equal opportunity regulations. The ingenuous organizational response was to create an office of equal opportunity (as a structural organization feature) without actually imbuing it with the power to affect anything…

Constraints on negotiated outcomes: The outcomes of negotiation processes are similarly wide ranging, but with some interesting constraints. First is a complementary constraint to the constraints on participating entities: if it is a negotiation, then there must be more than one reasonable outcome. Thus we elminate "cooperation without communication" [19] style situations where there is a clearly dominating outcome (clearly dominating for all participants and clearly known by them all). Such situations are not common outside of artificial systems because of course social factors can cause people and human organizations to stubbornly avoid "obvious rational" outcomes due to historical grudges, differences in belief systems, etc.

In previous definitions for IWMAS-98, several writers held forth that negotiation may be about avoiding conflicts. We would like to add that besides avoiding negative relationships to outcomes, negotiation may also be about achieving or taking advantage of positive relationships to outcomes.

Finally, the outcomes of negotiation processes can be usefully divided into three basic classes and meta-classes. Decisions must be made at three basic levels: what goals or objectives to achieve ; how to achieve them ; and when to schedule particular actions with respect to others [7]. Each of these levels also allows a subtley different level of meta-negotiation over the process used to come to these decisions: allowable future goals, future behaviors (as mentioned by Mike Huhns) and future timetables. We now discuss each of these three levels in turn.

First, agents may negotiate over appropriate goals or objectives (here, we typically use "goal" to denote some particular achievable state, and "objective" to denote a preference over states). This can also include goals of maintenance. Such negotiation processes may be couched in terms of relative goal importances, resource allocations, etc. Secondly, agents may negotiate over how to actually achieve or maintain some goal or satisfy an objective. Often the "how" is specified via constructed or intantiated plans or non-deliberative reactive behaviors, which can then become the object of negotiation. Certain actions or classes of actions may be allowed or disallowed, certain behaviors required in reaction to possible future stimuli. Finally, agents may negotiate over when actions should be carried out (either in an absolute sense or in relation to other actions).

Negotiation defined broadly becomes the central activity of a multi-agent system. With the exception of certain well-known and somewhat limited situations , negotiation is required at some point for almost all multi-agent behavior. As suggested, this extended abstract will focus on some opportunities and pitfalls for negotiation technology, and then discuss a bit about our experiences in the area of coordinating (arranging) agent activities.

1 Opportunities

With respect to opportunities, negotiation techniques provide several of interest. One of the most important is the recognition of multi-criteria decision problems and multiple characteristics that are important to individual agents and organizations. Agent preferences may arrise individually and from the organizations of which an agent is a member. The important observation is that these preferences are not necessarily the same for all agents, and that such non-identical preference structures provide the means for win-win negotiation situations. Such techniques are important in real-world applications (i.e. labor negotiation [30]) as well as abstract multi-criteria problems [33]. In complex problems, especially those involving people, preference structures may be too complex (or sometimes even inconsistant) to search for pareto optimal solutions and satisficing solutions may be OK. Another cause for settling for merely satisficing solutions is real-time problem solving ('soft' or 'hard') where negotiations must be completed by a (often non-negotiable) deadline, often resulting in agents that settle for a non pareto-optimal solution. One of the reasons for developing the TÆMS task representation framework was to capture and reason about the structure of complex multi-criteria problems [7, 8].

Another interesting opportunity (from an application standpoint) is the possibility of applying *automated* negotiation technology to standard interactions between human organizations and sub-organizations. At one level, this of course includes the very arms-length, electronic commerce types of technology that allow things such as automated purchasing via pre-negotiated contracts or automated auction or bidding systems. Such technology is of course quite useful, and likely to be easier to develop since it is simpler to abstract and formalize. However, not all inter-organizational relationships involve simple transactions. When transaction costs are high, long-term contracts may need to be negotiated [34]. The prototypical example would be the way an organization such as Boeing interacts with suppliers of office supplies, versus the way it interacts with suppliers of jet engines (the latter involving high transaction costs—it may take years for GE or P&W to develop a new engine for a future airplane). The added complication of high uncertainty in the future leads to fairly complex, contingency-filled contractual solutions. The use of automated negotiation technology at organizational boundaries—both inter-organization and intra-organization—is an interesting technology direction that will probably move over time from simple to more complex relationships. Technology for creating (temporary) organizations on-the-fly is already available (e.g. [6, 5]).

A final interesting opportunity is the possibility of dynamically chosen or constructed negotiation protocols. Previous work (of ours and others) has shown that most protocols will not be universally 'the best' in all situations, or situations that are dramatically changing over a period of time. Work on generic negotiation patterns [21] has provided templates for this kind of 'meta-negotiation' activity.

2 Pitfalls

There are many pitfalls to negotiation. The first is the problem of 'over-computation'. Many published algorithms are extremely complex and multi-stage, requireing a great deal of computation on the part of the agents involved. However, just because such an algorithm results in a guarenteed pareto optimal solution may not mean that it is a good algorithm for all situations. Often simpler approaches will work almost as well, and require much less computation. Sometimes this can be achieved by the way the tasks are structured initially (either by people, or through some meta-problem negotiation process). Several times we have shown that a more complex algorithm may do better in complex situations, but not in simpler or carefully-structured situations [10, 11, 13, 14, 26].

A second, well-studied pitfall is protocol manipulation. For certain classes of protocol, non-manipulable solutions have been developed (i.e. solutions where no agent can do better than a known algorithm). Sometimes, however, such solutions rely on external enforcement (usually monetary) [Ephrati][27] where the enforcement organization and protocols are not yet well-understood (which is perhaps just another area where work has not yet occurred). On the other hand, there is sometimes an over-reliance on the self-interest assumption on the part of agents. While this is an excellent and reasonable assumption in many cases (i.e. electronic commerce activities) it may not be a good assumption for intra-organizational activity arrangement. A good example are the multi-agent approaches to concurrent engineering, where agents may have thier own disparate viewpoints, but where giving up local utility in return for a higher overall evaluation is often desirable [Klien][22, 24]. An associated idea is work that builds on the idea of learning trust through repeated interactions [29].

Although simple, abstract negotiation protocols often have provably good properties, not much work has been done in integrating these protocols with organizational constraints on action, which might limit the actions of some participants more tightly than the full protocol (see for example Tambe [31]). A related issue is that of role integration—where an agent is playing several roles, perhaps also in several organizations. These sort of practical problems could be a pitfall for some simple approaches. Other typical pitfalls include the inability to deal easily with dyanamic, changing environments; large amounts of uncertainty in outcomes of current and future actions (see our earlier discussion of high transaction costs); inability to construct very complete views of the current situation.

3 Technology

The technology that we have been developing, Generalized Partial Global Planning (GPGP) [13, 26, 14], seeks to represent, choose and execute behaviors to respond to particular, perceived needs or opportunities for coordinated scheduling[1] of behavior. GPGP is a domain independent scheduling coordination approach that makes several architectural assumptions. Most important of these is that the agent represents its current set of intended tasks using the TÆMS task structure representation language [12]. These task structures are HTN abstraction hierarchies whose leaves are instantiated basic actions or "executable methods". Such an approach can be used both to describe human tasks[8, 24] and to program software agents [4, 15]. TÆMS allows the specification of dynamically changing and uncertain task characteristics that effect an agent's preferences for some state of the world, including tasks with hard or soft deadlines. A TÆMS specification also indicates relationships between local and non-local tasks or resources that effect these agent preference characteristics. An agent using the GPGP approach provides a planner or plan retriever to create task structures that attempt to achieve agent goals, and a scheduler that attempts to maximize utility via the choice and temporal location of basic actions in the task structure. Each GPGP mechanism examines the changing task structure for certain situations, such as the appearance of a particular class of task relationship, and responds by making local and non-local *commitments* to tasks, possibly creating new communication actions to transmit commitments or partial task structure information to other agents. The set of coordination mechanisms is extendible, and any subset or all of which can be used in response to a particular task environment situation. Initially, GPGP defined the following five coordination mechanisms based on Durfee's PGP:

1. Updating non-local viewpoints. Each agent detects the possible coordination relationships then communicates the related task structures.

2. Communicate results when they will be used by others.

3. Handling simple redundancy. When more than one agent wants to execute a redundant method, one agent is randomly chosen to execute it and send the result to the other interested agents.

4. Handling hard relationships (A must come before B) from the predecessor side. A is the "predecessor" task, B is the "successor".

5. Handling soft relationships from the predecessor side (if A is executed before B, the execution of B will be perhaps faster or will return better results, but it is not strictly necessary).

More recent work has developed simple coordination mechanisms for using mutually exclusive resources and for contracting (based on the well-understood approaches (e.g. [3, 28]). We have

[1] The term "planning" in the name is historical, from Durfee's PGP[16]. In the modern AI view of a continuum between planning and scheduling, both GPGP and PGP focus on the scheduling side.

applied different combinations of these mechanisms to problems in distributed vehicle monitoring, distributed data processing, hospital patient scheduling, and concurrent engineering.

3.1 Negotiation over Hospital Patient Care

Our view is drawn from a case study of an actual hospital [25]:

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

Furthermore, each test may interact with other tests in relationships such as enables, requires−delay (a slight variation on enables), and inhibits (a negative variation of the soft facilitates relationship). These task relationships indicate when the execution of one task changes the characteristics (here, primarily duration) of another task [12].

From this view of task structure, the hospital scheduling problem has these peculiar features:

- Tasks have no redundancy. Each test can only be done by a single ancillary department.
- The quality accumulation functions of non-executable tasks are always *min*. This is because here all the tests need to be done (TÆMS can however represent a wide range of quality accumulation functions).
- Unlike our previous domain models, here the final quality is not important. A test is either completed, or it is not.
- Different agents (representing different nursing or ancillary units) may use different utility functions when making local schedules. For example, the unit nurses may try to minimize the patient's stay time, while the ancillary staff tries to maximize equipment use and/or minimize setup times.
- Although TÆMS and GPGP handle them, there are no deadlines in this domain model.

Examining the hospital's current coordination structure is enlightening because it shows a mismatch between the structure and the current hospital environment. The current practice is for the nursing units to notify ancillaries of the requested tests and treatments, and for all ancillaries to schedule independently, sending an orderly to collect a patient whenever the scheduled time arrives. Of course, with no inter-ancillary coordination, there is no way to know if the patient will actually be there—they might be off having a different test performed. Unit nurses try to make sure the proper prerequisite tests (represented here by enablement constraints) are done first. While this structure seems sorely lacking when compared to the current environment, it may at one time have been a reasonable, low overhead arrangement. It may be that in the past doctors ordered fewer tests on less complex ancillary equipment with fewer or no interfering relationships between these technologies. Thus the current practice is adapted for a different task environment. In [14], we demonstrate that a simple algorithm, not much different from the current hospital scheduling practice, is not significantly different from our new algorithm when ancillary interrelationships are low or non-existent.

Although we can apply the five existing GPGP coordination mechanisms to the hospital patient scheduling problem directly, the hospital patient scheduling problem contains a task relationship that is not handled by any of the existing coordination mechanisms, namely, that the hospital patient is a mutually exclusive resource. To address this new relationship, first we must define it formally as a resource constraint in our task structure representation language (TÆMS [12, 32]) and then we define a new GPGP coordination mechanism to handle the new task relationship. We show how the existing practices can be captured using GPGP and *removing* some of the mechanisms. GPGP *with* our new resource-constraint mechanism outperforms existing practice on modern, complex procedures in two ways: increased patient throughput, and decreased patient stay time. Historically, hospital procedures were not as mutually interrelated as they are becoming today. We also demonstrate that the existing practice produces similar results to our new mechanism when few inter-unit task relationships exist.

3.1.1 The Simple Mutex Coordination Mechanism

In the hospital scheduling problem, some (but not all) tests can only be done with the patient physically being there. Thus a resource constraint problem arises, where the patient becomes a crucial non-sharable resource for different agents. A mechanism that can responds to this situation is not included in the five mechanisms initially defined based on PGP, and can not be created by simply combining some of those mechanisms. Therefore we describe a new resource-constraint coordination mechanism. The new mechanism uses a simple multi-round but not multi-stage negotiation process which is not optimal but has good "flow" properties since at least one agent is free to stop meta-level communication and begin domain work at each round [21]. Our experience is that in highly dynamic environments with large amounts of uncertainty, excessive time taken to find an optimal or near-optimal schedule is wasted [7]. Other assumptions that we make include that the agents are essentially cooperating with one another, and that their utility measures are comparable (here, we are usually dealing with the amount of time saved, which is a comparable measure).

When several agents try to use the same non-sharable resource at overlapping times, only one agent can actually get the resource and execute its work. The others who failed to get the resource waste this time unit and this effort. The idea behind the resource-constraint coordination mechanism is that when an agent intends to execute a resource-constrained task (i.e. the task is scheduled), it sends a directed bid of the time interval it needs and the local priority (expressed as the effect on local utility) of its task (we'll describe how this is computed later). 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, 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 won will keep its schedule and execute that task at the time interval it bid, and everyone else will mark this time interval with a *DON'T* commitment and never try to execute a related resource constrained task in it unless the owner gives it up. All the agents who didn't get their time intervals at this round will reschedule and bid again. The detailed process is as follows:

1. Get information about the bids given at the previous time unit, find who gets which time interval (i.e., the bid with highest priority), and make DO or DON'T commitments as appropriate. Also process retractions of commitments from agents that are giving up some time interval.

2. Reschedule. If the locally constrained task cannot be scheduled now, back to step 1. Otherwise, send out a new bid about the time desired and the corresponding utility.

3. Get information about the bids sent by others at the previous time unit, compare them to see if we get the time, and if not, mark the time taken by others and return to step 2. If we do get it, we also mark the time taken by others, and go to step 4.

4. Monitor the execution of the action in question. When we receive others' bids, we still mark the time intervals with the appropriate DON'T commitments for the scheduler just in case something goes wrong, the world changes, and we need to reschedule. If we find the task has significantly changed its execution time in the schedule, tell the others we give up the remaining time interval. If the task is taking longer, then go back to step 2 and bid again. If the task is finished, stop this process.

When using this mechanism to handle resource constrained problems, we will assume that each agent will be honest, i.e.. they will not cheat on the priority calculation. Since different agents may have different goals with respect to each other and with any global goals, the performance of such a system can be worse than that of a centralized system. In many domains such as hospital scheduling (or telescope observation scheduling [2]), We *cannot* centralize scheduling because it would take away the authority of each unit over the day-to-day control of its own activities. Our distributed approach matches with the existing human organizational structure. It also allows each unit to attempt to optimize slightly different measures, as may be used by administrators to evaluate human unit-level performance.

3.2 Negotiation Between Human and Computational Agents

One operational vision of concurrent engineering is one of a network of workstations that enable concurrent and distributed requirements assessment, design generation, and analysis /citedecker98q. It envisions that groups of humans and computational 'agents' will cooperatively create, resource, evaluate, execute, and monitor design processes or manufacturing plans. Work to date has not always directly addressed the problems and opportunities that arise from a truly distributed and cooperative environment.

In the rotorcraft body panel design domain [24], for example, the development of objectives, design alternatives, and component analyses (e.g. stress analysis, fiber placement simulations ofr manufacturability) are mixed initiative processes that can take place concurrently, and are carried out by the primary design engineer and his staff (stress engineer, cost engineer, weight engineer, manufacturabilty, etc.). As the human planner makes decisions to focus on certain enemy vulnerabilities he or she creates (concurrent) taskings for both staff members and various computational agents; as the staff uncovers certain information, that same information may impact back on the main planning task (revealing problems with certain requirement targets). This pattern of *interrelated task structures* that are partially shared by multiple agents with mixed initiative tasking appears in many commercial and military environments (air campaign planning, hospital scheduling, manufacturing scheduling, coordinating large software development efforts). These interrelationships provide not only potential problems to avoid (such as forgetting a required task) but also opportunities to take advantage of (such as doing a task needed by more than one other agent first, or doing a favor for another agent).

In general, each person and computational agent involved in such distributed cooperative work has a set of tasks before them that currently need to be done (identifying targets in several areas, collecting intelligence on each of several disparate targets, etc.). The tasks may come from a local initiative, may be assigned by a superior, or may just be subtasks needed to accomplish current goals. The tasks may be initially perceived as unrelated. In the Boeing MADEsmart/RaDEO domain [1, 24, 20] we developed agents to assist users in task selection called User Coordination

Assistant Agents (UCAAs) that keep track of a workstation user's current agenda of tasks and presents a possible schedule (ordering) of these tasks according to user- and domain-directed preferences. The idea is not to force a user into the schedule, but rather to provide the user with up-to-date, relevant information so that the user can make an informed decision. Such an agenda is not developed in isolation, but rather through a distributed coordination process using multiple coordination mechanisms triggered by the *coordination relationships* between the task structures of the different agents involved. We also developed an Agent Coordination Module (ACM) that can be linked-in to provide agenda management services for the coordination of computational agents in accord with the human agents' task-order preferences.

Such a set of tools allows the development of systems that can truly support cooperative work involving people at multiple workstations and computational agent assistants by helping to manage and organize the workloads of these agents. Certain tasks will be dependent on other tasks *and* be time critical, and such information for differentiating tasks could be lost in critical situations if only ad hoc coordinating mechanisms are used. It is useful to communicate more information than simply task assignments and task coordination relationships. For example, users and computational agents can request possible due dates for potential tasks before assignment (allowing both better time estimates and load balancing) and request *commitments* to complete certain time-critical tasks (in effect, supporting inter-agent negotiation protocols). Our approach of tying coordination mechanisms to the particular kinds of coordination relationships that exist in a particular task environment provides for a reusable tool foundation (the UCAA and its interactive human interface, the ACM and its interface to a computational agent) and a customizable set of coordination mechanisms for a particular application environment (since no single coordination mechanism would be useful across different environments with different task coordination relationships).

3.3 Negotiation Within an Agent

Negotiation processes can be useful within an agent as well [17]. For example,in real-time problem solving it is useful to separate real-time scheduling from decision making for reasons of efficiency, modularity and reusability. However, such a separation results in two, semi-autonomous components that need to work together to solve problems. Because they have different knowledge and different perspectives, these two components can disagree about what the agent should do. We have found that the interface between these components can usefully be modeled using negotiation. One example of a high level decision maker is a coordination module for an agent in a multiagent system that makes decisions about what information to communicate to other agents and when to communicate it. Of the various reasons for developing such a complex bidirectional interface, the reusability argument is the most clear. A scheduling component that attempts to find optimal method execution times for arbitrary task structures and evaluation criteria is useful in many domains, regardless of whether they have real-time or distributed problem solving components. We have demonstrated that such schedulers can be efficient [18, 32]. The other two reasons for a negotiation-style interface, efficiency and modularity, are closely tied. Efficiency is primarily the ability to search on, or attempt to optimize, specific criteria effectively. Defining a carefully delimited scheduling problem, and even limited search criteria, allows for the construction of a efficient scheduler. In fact, multiple schedulers can be constructed, each optimized to search (perhaps in parallel) under a different criteria (best quality, least violated commitments, earliest finish time). This leads directly to the modularity argument—that in general it is difficult to encapsulate all problem solving criteria into a single evaluation function, and deciding on the correct criteria is an evolving computational process (to be handled by a separate decision maker, we argue).

Note: The following is **not** *a comprehensive bibliography on negotiation, but just the papers cited in this extended abstract. I will try to extend this later, but I'm afraid I'm a little pre-occupied with my upcoming wedding :-).*

References

[1] T. Barrett, G. Coen, J Hirsh, L Obrst, J Spering, and A. Trainer. MADEsmart: An integrated design environment. 1997 ASME Design for Manufacturing Symposium, 1997. This paper is an extremely high level overview of the MADEsmart system concept for agent support (and multi-disciplinary optimization) for concurrent engineering, specifically, rotorcraft composite materials panel design. I've only included it for completeness; both the Obrst and Jha papers have more agent content. A posible application paper.

[2] J. Bresina, W. Edgington, K. Swanson, and M. Drummond. Operational closed-loop observation scheduling and execution. In *Proceedings of the AAAI-96 workshop on Theories of Planning, Action, and Control*, 1996. One of several papers on the telescope observation scheduling problem. I believe it to be an analog of the hospital scheduling problem. It is inherently distributed and non-centralizable, because the telescopes are physically distributed and belong to multipe organizations. Users need to propose observations (some of which may be limited to certain classes of telescopes) and all of the telescopes need to perform as many high-impact observations as possible (including reacting to dynamic conditions, e.g. weather). This paper does not talk about the distributed problem in particular (which I have discussed with one of the authors at a AAAI spring symposium). An application of negotiation technologies.

[3] R. Davis and R. G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, January 1983. The original Contract Net paper. Interestingly, (although I haven't read it in a while) it proposes the view as a type of "content-directed procedure invocation", i.e., a way for procedures to decide if they can and should execute (perhaps, a long way from the agent-oriented point of view). To follow on, compare this to Sandholm's TRACONET work and other papers working out some of the contract net details (e.g. whether to reserve resources for a bid until the accept/reject is recieved). Historically important paper.

[4] K. S. Decker, A. Pannu, K. Sycara, and M. Williamson. Designing behaviors for information agents. In *Proceedings of the 1st Intl. Conf. on Autonomous Agents*, pages 404–413, Marina del Rey, February 1997. This paper is about an internal architecture and set of behaviors for what would now be called "information extraction" agents. The idea is to make these full of features, but to limit the amount of new programming needed for each (so one can quickly create many of these full-featured, low-level worker agents). The reusable behaviors included the start of some very simple negotiation behaviors (i.e. advertising a service, "negotiating" a request in the sense of "I can't provide 15 minute-interval updates, how about 30 minute updates?", matchmaking and brokering (see below)).

[5] K. S. Decker, K. Sycara, and M. Williamson. Middle-agents for the internet. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 578–583, Nagoya, Japan, August 1997. A detailed look at simple third party service negotiation, organized around a concept of privacy (who knows the service advertisements or service results? who knows individual information needs?). We examine some of the simplest possible solutions and analyze them for throughput and ability to adapt to changing situations (like agents leaving

in an open system). The ideas are based from sources like the original KQML paper and other agent facilitator work (e.g. Wiederhold), through to blackboard systems (from a service provider privacy standpoint, the opposite of yellow-pages/matchmaking systems).

[6] K. S. Decker, M. Williamson, and K. Sycara. Modeling information agents: Advertisements, organizational roles, and dynamic behavior. In *Proceedings of the AAAI-96 Workshop on Agent Modeling*, 1996. AAAI Report WS-96-02. A short paper describing one way to go for service advertisements: starting proposals for service negotiation. For a simple advertisement, any request meeting the advertisement can be implicitly considered an acceptance of the proposal. For more complex advertisements, this can only be a starting point providing, for example, a range of service characteristics (i.e. price, reliability, completeness) that may need to be negotiated in detail on an individual request basis. Sycara's group at CMU is now working on a much more formally detailed advertising model (for a single well-defined class of services). Also see the take on advertising by Sarit Kraus and V.S. Subrahmanian, which is oriented toward describing the interface to the service (number and type of arguments needed in a request).

[7] Keith S. Decker. *Environment Centered Analysis and Design of Coordination Mechanisms.* PhD thesis, University of Massachusetts, 1995. `http://dis.cs.umass.edu/~decker/thesis.html`. This is the definitive source for descriptions of GPGP and TÆMS although TÆMS has been extended by both myself and other people at UMass (e.g. Alan Garvey and Tom Wagner) and some of those specifications are still in press at this time. There is also a fairly nice review of social organization research oriented specifically towards DAI, MAS, and coordination problems. This work followed from Ed Durfee's PGP work in the DVMT and Frank von Martial's work on task relationships. The core ideas here with respect to negotiation are primarily at the planning and scheduling level: the TÆMS representation of some task environment (representing what agents individually believe the problem structure is, versus what it *actually* is) and the GPGP approach to coordination, which is to use an extendible set of coordination mechanisms that recognize and react to certain potential pitfalls and opportunities in each agents' local TÆMS view of the tasks at hand. The fact that TÆMS is carefully specified (based on a state-based model of computation) allowed me to quantitatively analyze several situations (i.e. the DVMT, hospital scheduling) that formerly could not be carefully analyzed. The GPGP mechanisms in the thesis are derived specifically from those needed for the DVMT problem, but then we applied them to other real and synthetic domains. These mechanisms included communication of pieces of an agent's private view of the problem, various types of commitments to action (or inaction, or to particular approaches to problem solving), and reasoning about deadlines.

[8] 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. This book chapter gives a simple introduction to TÆMS as a modeling language, and some examples of modeling hospital scheduling, concurrent engineering, airport resource management, pilot's associate, and internet information gathering. Although it does not contain all the newest info on TÆMS it is often the first thing I have a student read before we discuss the details of the TÆMS modeling language. In a negotiation context, it presents several domains in need of negotiated task coordination, and a language which expresses the differences in what the agents perceive. Unfortunately all these examples are cooperative (and TÆMS is not limited to only cooperative domains—it just happens to be an interest of mine). There is a prisoner's dilemma example in my thesis.

[9] Keith S. Decker. Coordinating Human and Computer Agents. In W. Conen, G. Neumann, editors, *Coordination Technology for Collaborative Applications - Organizations, Processes, and Agents.* LNCS #1364, pp. 77–98, Springer-Verlag, 1998. This book chapter discusses negotiation for coordination in the context of concurrent engineering systems. Note that this is a different problem than negotiating a *design* concurrently. Instead we are trying to support the management of many people and software agents working on several shared tasks in a dynamically changing open work environment.

[10] Keith S. Decker and Victor R. Lesser. An approach to analyzing the need for meta-level communication. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 360–366, Chambéry, France, August 1993. This paper examines a DVMT problem earlier examined by Corkill, Lesser and Durfee. By using the more formal TÆMS specification as a modeling language, we are able to find statistical regularities and make a more comprehensive, predictive analysis of certain coordination techniques. In particular, this paper shows that some complex negotiation techniques for coordination are not always useful (and shows what domain features predict this result). At another level, this paper is about a experimental *methodology* that goes beyond simply saying negotiation protocol X is better than Y.

[11] Keith S. Decker and Victor R. Lesser. A one-shot dynamic coordination algorithm for distributed sensor networks. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 210–216, Washington, July 1993. This is a follow-up to the previous paper "An approach to analyzing the need for meta-level communication". Using the same basic DVMT model (laboriously derived in that paper :-) this paper examines some fancier negotiations, voting mechanisms, for the classic DVMT problem, and provides a similar predictive analysis. The idea is that if you have, say, 4 coordination mechanisms and a good predictive model, you can create a hybrid algorithm that chooses from the 4 based on expected value with respect to the current problem solving situation.

[12] Keith S. Decker and Victor R. Lesser. Quantitative modeling of complex computational task environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 217–224, Washington, July 1993. This was the original TÆMS paper. I would use instead as a starter paper "Task environment centered simulation", and then Chapter 3 of my thesis (until the upcoming new TÆMS specification is available). This paper is too short on examples I think to be understandable to the casual reader.

[13] Keith S. Decker and Victor R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 73–80, San Francisco, June 1995. AAAI Press. A longer version is available as UMass CS-TR 94–14. Also reprinted in M. Huhns and M. Singh, editors, Readings in Agents. Morgan Kaufmann,1997. This is the orginal GPGP paper. See also the description of my thesis. The planning "P" in GPGP and PGP is a slight misnomer, as modern AI planning people would probably think of this more on the scheduling end of the planning/scheduling continuum. GPGP went beyond Durfee's PGP work by 1–seperating out individual coordination mechanisms (in this paper, only mechanisms needed for solving the DVMT problem, but later other mechanisms as well), 2–expressing them in a general language not tied to the DVMT, 3–expanding them to deal with hierarchical tasks, heterogeneous agent capabilities, and real-time deadlines. Lots of work has follwed on from this: Garvey and Wagner's works on a real-time schedulers for such systems, Nagendra-Prasaad's work on learning appropriate coordination mechanisms, my students Liu

and Shao's work on new coordination mechanisms for mutual exclusion (hospital scheduling domain) and resource constraints (information gathering domain).

[14] Keith S. Decker and Jinjiang Li. Coordinated hospital patient scheduling. In *Proceedings of the 3rd Intl. Conf. on Multi-Agent Systems*, Paris, 1998.

[15] Keith S. Decker and Katia Sycara. Intelligent adaptive information agents. *Journal of Intelligent Information Systems*, 9(3):239–260, 1997.

[16] E.H. Durfee and V.R. Lesser. Partial global planning: A coordination framework for distributed hypothesis formation. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1167–1183, September 1991.

[17] Alan Garvey, Keith Decker, and Victor 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.

[18] Alan Garvey and Victor Lesser. Representing and scheduling satisficing tasks. In Swaminathan Natarajan, editor, *Imprecise and Approximate Computation*, pages 23–34. Kluwer Academic Publishers, Norwell, MA, 1995.

[19] M. R. Genesereth, M. L. Ginsberg, and J. S. Rosenschein. Cooperation without communication. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 51–57, Philadelphia, PA., August 1986.

[20] K.N. Jha, A Morris, E. Mytych, and J. Spering. Agetn support for collaborative design. In *Proceedings of the DETC98: ASME Computers in Engineering Conference*, Atlanta, GA, September 1998.

[21] Brigitte Lâassri, Hassan Lâassri, and Victor R. Lesser. Negotiation and its role in cooperative distributed problem solving. In *Proceedings of the Tenth International Workshop on Distributed AI*, Texas, October 1990. Also COINS TR–90–39.

[22] S. Lander and V.R. Lesser. Negotiated search: Organizing cooperative search among heterogeneous expert agents. In *Proceedings of the Fifth International Symposium on Artificial Intelligence Applications in Manufacturing and Robotics*, December 1992.

[23] J.W. Meyer and B. Rowan. Institutionalized organizations: Formal structures as myth and ceremony. *American Journal of Sociology*, 83:340–363, 1977.

[24] L. Obrst, M. Woytowitz, D. Rock, S. Lander, K. Gallagher, and K. S. Decker. Agent-based integrated project teams. In *Proceedings of the EIM97 Engineering Information Management Symposium of the ASME Design Engineering Technical Conferences*, Sacramento, CA, September 1997.

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

[26] M.V. Nagendra Prasad, K. S. Decker, A. Garvey, and V.R. Lesser. Exploring organizational designs with TÆMS: A case study of distributed data processing. In *Proceedings of the Second International Conference on Multi-agent Systems*, Kyoto, Japan, December 1996.

[27] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers.* MIT Press, Cambridge, Mass., 1994.

[28] Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 256–262, Washington, July 1993.

[29] S. Sen and A. Biswas. Effects of misconception on reciprocative agents. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 430–435, Minneapolis, May 1998. ACM Press.

[30] K. Sycara. Multi-agent compromise via negotiation. In M. Huhns and L. Gasser, editors, *Distributed Artificial Intelligence*, volume Volume 2. Pittman, 1989.

[31] M. Tambe. Agent architectures for flexible, practical teamwork. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Providence, July 1997.

[32] T. Wagner, A. Garvey, and V. Lesser. Complex goal criteria and its application in design-to-criteria scheduling. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, Providence, July 1997.

[33] M.P. Wellman and J. Doyle. Modular utility representation for decision-theoretic planning. In *Proc. fo the First Intl. Conf. on Artificial Intelligence Planning Systems*, pages 236–242, June 1992.

[34] Oliver E. Williamson. *Markets and Hierarchies: Analysis and Antitrust Implications.* The Free Press, New York, 1975.