Modeling Information Agents: Advertisements, Organizational Roles, and Dynamic Behavior

Keith Decker and Katia Sycara and Mike Williamson
The Robotics Institute, Carnegie-Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
(decker,sycara,mikew)@cs.cmu.edu

Abstract

One of the most important uses of agent models is for problem-solving coordination. Coordination has been defined as managing the interdependencies between activities, or pragmatically as choosing, ordering, and locating actions in time in an attempt to maximize a possibly changing set of decision criteria. Coordination activities include not only localized agent interactions over specific problems, but also longer-term agent organizations that can support current and future problem-solving activity. Models that support coordination can be divided into three classes: models of other agents' current intended actions, schedules, and/or plans; models of other agents' objectives (desires, goals); and models of other agents' capabilities.

This paper will focus on the longer-term models of capabilities that agents need in order to organize effectively to solve problems. Such models deal with an agent's capabilities and long-term commitments to certain classes of actions. Besides discussing the models themselves, we will discuss the basic behaviors agents should use to construct, communicate, and update these models. Finally, we will discuss how these models can support several different organizational forms such as economic markets, Wiederholdian federations, and bureaucratic functional units. The work will be presented in the context of an implemented agent architecture for Internet information gathering in dynamic domains such as financial portfolio management (called WARREN).

Introduction

One important reason that software agents build models of other agents is so that they can choose, order, and time their actions with respect to the actions of others. Such choices are made in the context of an agent's objectives[1] and its knowledge of the objectives of other agents. Examples of such reasoning include:

- Simulated wingmen acting to stay in formation with their wing leader, and acting to avoid bad outcomes from confrontations with enemy pilots (Tambe 1995)
- Agents reasoning about the likely utility of possible decision outcomes from their own and other agents' points of view so as to make a local decision on a particular action(Rosenschein & Genesereth 1985; Gmytrasiewicz, Durfee, & Wehe 1991)
- Agents reasoning about what to say and when in carrying out a conversation or negotiation (Chu-Carroll & Carberry 1995; Sycara 1989)
- Agents reasoning about how to choose between, order, and temporally locate sequences of actions that arise dynamically during problem solving (Durfee & Lesser 1991; Decker & Lesser 1995)

Most of this work has focussed on representing in the agent model (1) the particular actions, schedules, and/or plans of other agents and (2) the objectives of other agents. However, equally important is (3) the capabilities of other agents. For example, in the tactical air domain the doctrine of the various combatants, and the physical capabilities of the aircraft, become a constraint on potential actions and objectives (compare also limited vehicle velocity, acceleration, and turning radii in the DVMT (Corkill, Lesser, & Hudlická 1982)). Some conversational models take into account limited human short-term memory (thus causing reminder-utterances to be produced) (Walker 1994). Some models of non-local schedules and plans include capability information (e.g., that agent 1 can execute an action twice as fast as agent 2) (Decker & Lesser 1995). The major issue addressed by this paper is how to represent the capabilities of information agents in a way that can be utilized for various organizational and plan-coordination purposes. Furthermore, our solution is constrained by our chosen application domain, open system multi-agent information gathering.

This paper will first briefly describe information agents, their basic behaviors, and the constraints placed on capability modeling by the environment. Next, we will define a communication model, overlaid on KQML, within which we can define what it means to advertise an agent's capability. We will propose an extensible, KQML-aligned, capability representa-

[1] We'll use the word *objective*, rather than "goal" or "desire", to indicate a potentially complex specification of a desired outcome and its associated multiple-attribute utility characteristics.

tion and a corresponding basic information agent behavior for advertisement. Finally, we will show how basic capability advertisement can be incorporated into several popular agent organizations, including markets, federations, and functional units.

Information Agents

The domain we have been working with is that of open system multi-agent information gathering (Sycara & Zeng 1995; Oates *et al.* 1995). Such multi-agent systems can compartmentalize specialized task knowledge, organize themselves to avoid processing bottlenecks (using models of other agents' objectives), and can be built expressly to deal with dynamic changes in the agent and information-source landscape (new and changing agent capabilities). Such systems may comprise many different types of agents, including *information agents* that are closely tied to a source or sources of data. We will be discussing capability advertisement for simple and multi-source information agents.

The main function of an information agent is to process intelligently and efficiently one-shot, periodic, and information monitoring requests. These requests come externally from other agents; the information used to fulfill these requests comes from arbitrary external information sources. Typically, a single information agent will serve the information needs of many other agents (humans or computational agents). An information agent is quite different from a typical WWW service that provides data to multiple users. Besides the obvious interface differences, an information agent can reason about the way it will handle external requests and the order in which it will carry them out (using its models of others). Simple information agents can carry out long-term interactions that involve monitoring for particular conditions, as well as simple information updating. More complex multi-source information agents deal with the problems of dynamically combining information from different simple information agents in ways that provide more complete information, balance the load on the underlying sources, and add extra robustness and timeliness.

Information agents are built on top of a common set of reusable basic behaviors. By "behaviors", we mean the execution of partially-ordered sequences of basic actions. By "reusable", we mean only that the behaviors are specified in terms of domain-independent abstractions. To instantiate a new simple information agent, one needs to define a database schema (what information this agent presents to the world) and develop a small piece of site specific code to handle a portion of the external data access. We have implemented information agents and these shared, basic behaviors, and have tested them experimentally on the WWW in several domains: tracking stock prices, extracting news stories, retrieving fundamental historical SEC data, monitoring airfares, and KQML matchmaking. Basic behaviors for simple information agents include answering one shot and periodic queries, monitoring for some pattern or change, polling for messages, maintaining and rebuilding the local DB cache, self-cloning to improve response times and balance loads, and advertising an agent's capabilities.

Because the environment that these agents are in is a dynamic and open one, where new sources and agents with new capabilities are constantly coming and going, we cannot use the traditional solution to the capability modeling problem—assuming that all agents have the same, or predetermined capabilities. Instead, new agents must be able to communicate thier capabilities through thier advertisements (regardless of to whom these advertisements are sent, or how they are ultimately used in the organization—see the next-to-last section).

Modeling Other Agent's Objectives and Capabilities

Next we will discuss the set of communication principles that determine how agents know about one another's capabilities, constrain how and when an agent accepts requests, and constrain the behaviors used by the agent to carry out those requests. How can we guarantee that information agents act in ways that are predictable and useful enough for them to take part in larger multi-agent systems and organizations? We, along with others believe that commitments of various types are the key to coordinated multi-agent behavior (Jennings 1993; Cohen & Levesque 1990; Decker & Lesser 1995; Rao & Georgeff 1995). KQML, however, has no explicitly commisive performatives. We resolve this by assigning commisive semantics to the KQML ADVERTISE performative. This section will first discuss a generalization of the large set of KQML communicative acts that will allow us (and our agents) to more succinctly represent and reason about *classes* of communicative acts, rather than needing to handle every performative as a separate case. We will then discuss the ADVERTISE performative and an extensible agent capability model, itself based on KQML.

Communicative Acts and Parameters

Rather than either discussing (and reasoning about) every performative individually, we can group them (as suggested by speech act theory and (Cohen & Levesque 1995)) into three classes: assertives (TELL, REPLY, ERROR, ADVERTISE (explicitly) ...), directives (ASK-ONE, ASK-ALL, STREAM-ALL, MONITOR, ACHIEVE ...), and commisives (ADVERTISE (implicitly), but perhaps some day things like COMMIT, PROPOSE, AGREE, POSTPONE, ACCEPT-ROLE, JOIN-ORGANIZATION, etc.) An interesting attribute of KQML is that some parts of the message content are separated out into parameters (:sender, :receiver, :reply-with, :ontology, :language, ...). Because information agents deal with potentially time-sensitive information, we have added two more parameters, :deadline and :period. Deadline used

by itself refers to an absolute time by which a request must be serviced; soft deadline specifications are possible. Period refers to how often a request must be re-fulfilled. When period and deadline co-occur in a directive the deadline is assumed to be relative to any change in the directive's results. Finally, at least for the purposes of capability advertisement (see the detailed description later), we can view all of the various query directives (ASK-ONE, ASK-ALL, STREAM-ALL, GENERATE, ...) as variants on a basic ASK directive and a directive parameter ":report-action" that describes what to do with the query results (send the first, send them all in a list, send them all in individual messages, hold them for me to request later, ...).

Modeling other Agents' Objectives

In tracking other agents' objectives, an information agent sends and receives messages about both objective creation and objective cessation (modifying objectives is also possible, although not presently supported outside of cessation of the old objective and creation of a new objective) Both objective creation and cessation may be assertive (telling another agent a belief) or directive (requesting another agent to do something: achieve an objective, answer a query in a certain way etc) speech acts. Here is a list of these basic communicative classes and their meaning:

Assertion of objective creation: An information agent may assert to another agent that it has a objective, and in particular ADVERTISE asserts that an agent has an objective to service the objectives of other agents if they fall within certain constraints (i.e., to answer any queries on its database). This is an implicit commisive communication. When an information agent advertises its database by sending the database specification to, for example, some facilitator, it is making an implicit commitment to accept requests on the class of query objectives valid for that database. Any other agent can then assume that if it sends a query request to that agent, the agent will accept that request and work for its achievement until it is either achieved or the accepting information agent believes it cannot be achieved.

Request for objective creation: An agent may request that an information agent adopt the objective of answering some query Q, e.g., "Tell me the price of IBM every 10 minutes". This is done via the standard KQML directives. Information agents are free to reject requests that do not match their previous advertised commitments.

Assertion of objective dissolution: An information agent may assert that a objective, accepted from another agent, is being removed. This assertion normally carries with it a reason—typically, the query was malformed, the agent is unable to meet it's implicit commitment because it is too busy, or the agent is shutting down due to circumstances beyond its control. "I can no longer provide the price of IBM every 10 minutes due to a shutdown for preventive maintenance". Currently, there is no specific KQML performative for this communicative act, and our agents use ERROR/SORRY with an appropriate :in-reply-to parameter to indicate which objective is being dissolved.

Request for objective dissolution: An agent my request that an information agent give up a objective previously requested by the agent. "Cancel the objective of telling me the price of IBM every 10 minutes". Currently, there is no specific KQML performative for this communicative act, so we use UNMONITOR.

Information agent behaviors that deal directly with objectives from other agents, then, must obey these implicit commitments. They can do this because they are built on an agent architecture that provides planning and scheduling mechanisms that detect or predict action failures.

Modeling Other Agent's Capabilities

We believe that a very useful model of agent capabilities is one that describes the constraints under which an agent will accept another agent's objectives. We will call such a description a *service* description. The key to an extensible agent service description (advertisement) language are the KQML parameters—a great deal of the necessary capability descriptions are in fact constraints on service request parameters. An agent capability description, or service advertisement, consists of three parts: basic service characteristics, service modifier characteristics (tied to KQML parameters), and service specific characteristics. The idea is that the first two sets of characteristics are given fixed semantics irrespective of the particular service being advertised, but that different types of services can also have service-specific characteristics, defined on a service-by-service basis (akin to the :content parameter of a KQML message). Matchmakers, Facilitators, Brokers, Market-Makers, and default individual agent behaviors can be defined to operate only on the fixed portion and avoid the service-specific portion. Service-specific capabilities for well-known capabilities, such as the query-answering capabilities of information agents, will also be common knowledge (i.e. used in default agent behaviors and exploited by some types of message-routing agents). This section will discuss both the fixed capability model and the service specific model used by all of our information agents.

Basic Service Characteristics Basic service characteristics deal with information applicable to any type of service but not related to specific request/objective constraints. We believe that agents will attempt to make boundedly rational decisions in environments containing considerable uncertainty and where there are multiple, changing decision criteria. Such high-level decision criteria include cost, time, precision, certainty, reliability—all basic service characteristics. Basic service characteristics therefore include, but are not limited to:

pricing info: Service cost. This includes the concepts of a basic price, price modifier additions, and bulk or custom pricing flags. Price modifier additions map from service modifier values (i.e. KQML message parameters as expanded in the previous section) to price changes. For example, an agent may charge a higher price to provide higher levels of service (like very short periods). Bulk pricing refers to the practice of providing n requests for a fixed cost (used by some stock ticker providers, for example). Custom pricing indicates that cost varies on a case-by-case basis (agents might, for example, request a bid from the agent on a particular objective).

duration info: Service duration information, possibly including ranges and averages.

reliability: Information about the likelihood of service failures.

Service Modifier Characteristics Service modifier characteristics refer to constraints placed on the service modifiers (KQML message parameters).

sender: constraints indicate organizational authority/power relationships (e.g., an agent will only accept objectives from certain other agents)

receiver: nominally, the agent itself, *but* some agents may work only through proxies/secretaries/front offices/brokers. Thus a capability advertised by agent X with the receiver slot constrained to agent Y means that agent X can only be contacted via Y.

ontology, language, period, deadline, report-action: constraints on potential values for each of these KQML message parameters can be given individually.

Service Specific Characteristics for Information Agents An information agent's local database is defined in terms of an *ontology*, a set of *attributes*, a *language*, and a *schema*. The database ontology links concept-names to their data types and domain-specific meaning (and must match on any incoming KQML message).[2] Database *attributes* are meta-information stored about a field in a database record. By default, information agents will keep track of two attributes: timestamp and previous value. The timestamp indicates when the field value in the record was last updated. Note that when a local agent database is formed from information gathered from multiple external information sources, the timestamps on every field in a record may be different. The previous value is the value the field had before the last database update. No absolute temporal relationship between the current value and previous value can be assumed—only the relative temporal relationship ("before").[3] The database *query language* specifies a pre-determined format for the KQML "content" slot. The "simple-query" query language is a set of predicate clauses on field values and attributes, joined with an implicit "AND".

The local database is constructed from a database definition language description. This description also serves as a way to describe the information content a particular information agent offers to other agents. The important parts of the description, defined for each field in the database, are:

concept-name: the column or field name, the term referring to what the value in the database represents. A concept-name plus the ontology implies a data-type (string, integer, float, data/time, enumerated, etc.)

selectable flag: A field is flagged "selectable" if it possible to use that field to select records in a query. If the external database is a full-fledged modern relational DB, then usually all fields will be flagged "selectable". However, many of the databases available over the WWW do not allow record selection by every field, and so the agent needs to include this in its specification.

primary key: A field is flagged as the primary key if every record in the database must differ on that field. Every database definition must have a primary key. This field does not have to be selectable (or advertised). If necessary, it can be internally generated.

predicates: *optional*. If present, it limits the predicates that can be used in queries on this field. If absent, all of the natural predicates associated with the data-type for the concept-name can be used. Typically, a key field might limit the operators to EQUAL only.

range: *optional*. Gives information on any range limits associated with the field in question. If not present, the natural limits for the field's concept-name are assumed.

For example, the Security APL server is an Internet information source that has stock prices. Our Security APL information agent advertises:

```
(ADVERTISE
  :SENDER secapl :RECEIVER matchmaker
  :ONTOLOGY advertisement
  :content
    (SERVICE security-apl
      :PRICING 0
      :DURATION (5 seconds)
      :RELIABILITY .89
      :SENDER :ALL
```

[2] This paper will not discuss ontological representation or reasoning, see for example (Gruber 1993; Collet, Huhns, & Shen 1991).

[3] This restriction can be removed either by adding another basic behavior to an information agent that creates in effect a new "value-history" attribute, or by simply adding value-histories to the local database definition itself. The first alternative is more conservative of space.

```
:RECEIVER secapl
:LANGUAGE simple-query
:ONTOLOGY stocks
:PERIOD   (> 5)
:DEADLINE (> 10)
:REPORT-ACTION (one all stream)
:TYPE answer-query
:CONTENT
  (:ATTRIBUTES previous-value timestamp
   :SCHEMA
     (symbol :SELECTABLE :PRIMARYKEY
       :PREDICATES (eq))
     (company :SELECTABLE
       :PREDICATES (eq =~))
     (exchange :RANGE (NYSE AMEX NASDAQ))
     (reference-date :RANGE (- *now* 15))
     (price )))
```

Advertising Behaviors

Along with the basic behaviors described earlier to deal with one shot, periodic, and pattern-monitoring queries, message polling, local database cache maintenance, and cloning, simple and multi-source information agents use a shared basic advertising behavior, based on their local database schema as described in the previous section. Figure 1 shows the task structure which results from the basic planner reduction of the "advertise" task, as shared by both types of information agents.

The reduction is best understood in two parts: the first is the initial communication to the matchmaker, the second are the actions associated with the "shutdown" of the advertise task (Make Un-Advertisement and the SendKQML on the lower right). The task structure is expressed in a hierarchical task network planning formalism developed for information gathering agents that focuses on representing the information flow in a plan, and allows the representation of sophisticated control relationships (such as loops and periodic[4] actions). An important focus of our current (Williamson, Decker, & Sycara 1996) and earlier work (Decker & Lesser 1993) is that control relationships are *derivative* of other, more basic relationships such as required information flows, effects of one task on the duration or result quality of another, etc. The planner fleshes out these more basic relationships; the agent's scheduler uses this information to produce a schedule (or schedules in a dynamic multi-criteria decision-making situation (Garvey, Humphrey, & Lesser 1993)) that selects, orders, and locates in time specific instantiated actions. More information on the planner and this formalism can be found in (Williamson, Decker, & Sycara 1996).

The three actions "Make Advertisement", "Get Matchmaker Name", and the topmost instance of "SendKQML" are involved in sending the advertising message. Both "Get Matchmaker Name" and this instance of "SendKQML" are periodic. All three tasks have an initial deadline of "as soon as possible". "Make Advertisement" constructs the KQML advertisement message content (using the agent's local database schema plus execution information gathered and persistently stored from previous invocations) and provides it to SendKQML. The typical first reduction for "Get Matchmaker Name" is to use a predefined name; alternate reductions involving persistent memory or more complex network level protocols are possible.[5] The matchmaker name is supplied as the RECEIVER of the KQML message, and the REPLY-WITH field is supplied by the planner at reduction-time. If no matchmaker currently exists or some other network error occurs, the SendKQML signals a "DOWN" outcome, which provides a new signal to Get Matchmaker name, and the two tasks are rescheduled (they are periodic) and rerun. In general, the planner strives to produce structures such that agents can appear on the scene at any time and in any order (as well as disappear, which see next).

The two action instances "Make Un-Advertisement" and the second, lower right SendKQML instance comprise the *shutdown* actions for this task. A task is shutdown whenever:

1. The planner removes the current reduction (because, for instance, it has failed). This would not typically happen for advertisement, but does for other tasks.

2. The agent itself intentionally (via a "Termination" action) or unintentionally (unrecoverable error/signal) goes offline.

Shutdown actions are placed on both the regular and a special shutdown scheduling queue. Both actions are non-periodic and have a deadline of "only execute if there's nothing better to do". Actions on the shutdown queue are architecturally guaranteed to execute at least once, so in particular, the "Make Un-Advertisement" action will either execute during normal processing when the agent would otherwise be idle, or during shutdown if the agent never managed to have a second to spare. The SendKQML that actually passes the advertisement retraction on to the matchmaker has two extra enablement conditions: first, that the initial advertisement actually completed without error, and secondly that the task is being shutdown.

This basic advertising behavior is shared (reused) for all information agents (simple and multi-source). Other types of task agents also share this behavior, except that the tasks "Make Advertisement" and "Make Un-advertisement" are expanded differently for agents that are not advertising a local database.

[4] Technically, what we are calling here informally "periodic" are formally "distance-constrained" or "having a max repeated invocation separation constraint".

[5] In our system, the matchmaker is the only agent with a known name. The default matchmaker name can be changed by a Unix environment variable, thus our group can have agents working in multiple logical agent namespaces even though they share a small set of physical processors.

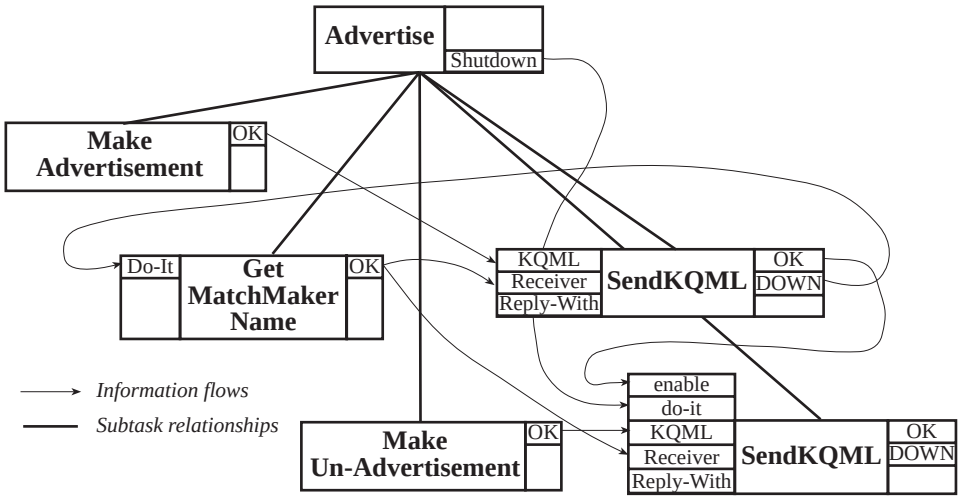

Figure 1: One planner task structure reduction for the "advertise" task.

Organizing Based on Capability Models

The previously described agent capability model and accompanying advertising behavior are very useful for forming several different agent organizations. These organizational possibilities include organizing as a market (agents choose the lowest-priced service to do the queries), as a Federation (agents defer all choice to a Facilitator Agent), or a functional unit (a Manager multi-source information agent speaks for a set of simple information agents that 'work' for it).

The simplest organization supported is the dynamic, unstructured, and uncoordinated team. Agents who request an action or information figure out at that point if the query can be answered and by whom, choosing an action- or information supplier randomly if there is more than one. For one-shot, periodic, or change-monitoring queries, this simple behavior is implemented via a task reduction shared by all information, task, and interface agents for the task "AnswerQuery". The shared behavior examines the *target* query and constructs an appropriate *matchmaker* query. The matchmaker query would check for any agent that has advertised to answer queries in the target language and ontology, with the appropriate field names.[6] An agent is chosen (randomly) from this list, and the target query is wrapped in the appropriate KQML performative and sent on. Failures (either network failures, or in the case of monitoring and other long-term queries the failure of the remote agent) cause the agent to go back to the matchmaker, looking for a new agent. With sufficient redundancy and/or a quick restart for system failures, the resulting system is fairly robust. However, it is not particularly good at balancing the load on redundant information sources, dealing with multiple ontologies or queries that cannot be answered by a single agent, or providing *predictably* robust service (as opposed to merely being able to recover from errors. Different organizational forms can better provide some of these characteristics.

For example, market coordination mechanisms (Wellman 1993) can provide better load balancing as well as more sophisticated services that cannot be provided for free. One way to implement this is to more carefully peruse the list of potential agents returned by the matchmaker query, paying attention to the pricing and reliability information. Rather than using one-shot matchmaker queries, agents can subscribe to matchmaker information streams about their important queries, so as new information sources come on-line, the agent can take advantage of possible lower prices and/or higher reliability (transaction cost considerations aside). The basic advertising behavior could be broadened so as to periodically (or under certain trigger conditions) update the agent's advertisement (i.e., reliability and pricing information that may factor in load). This brings with it new pluses and minuses; our purpose here is only to argue that we provide the basic support for such an organization.

Federated organizations (Wiederhold, Wegner, & Cefi 1992; Genesereth & Katchpel 1994; Finin *et al.* 1994) provide a different set of organizing principals. In these architectures what we have been describing as a fairly limited matchmaker becomes a *facilitator*, taking on the new behaviors of brokering, mediation, and translation (between ontologies). For example, in a brokered organization, the AnswerQuery task is reduced to one in which the facilitator (matchmaker) takes on the target query, and accepts responsibility for finding who can answer it, forwarding the query to them, and forwarding the reply to the original agent. By centralizing communication, such organizations allow easier load balancing and the provision of services such as ontological trans-

[6] Remember, these names are fixed by the ontology—this is an important service provided by the information agent abstraction over the true underlying data.

formations or simple multiple query integration, but place a great extra load (and responsibility) on the facilitator agent.

Traditional human bureaucratic functional units provide another possible organization. A multi-source information agent acts as a manager for agents with similar data sources and ontologies, such as a stock ticker manager multi-source information agent and a slew of individual stock ticker simple information agents (Security APL, Galt, DBC Online, Lombard, etc.) with partially overlapping schemas. When the manager comes on-line, it recruits the others as employees. These agents then un-advertise from the main matchmaker, and the manager becomes the sole entry point for queries to the new corporation. The manager acts as a KQML broker or recruiter for this limited set of queries, allowing tight control over load balancing, greater service reliability, and a potentially wider range or services (i.e. queries that cannot be answered by a single agent acting alone).

Current & Future Work

One important are of future work is in expanding the advertisement ontology to new classes of service. One new class of service we need for our application is graphical web page construction, which is currently done by two of the WARREN agents. Currently, these agents advertise thier capabilities as if they were simple information agents. For example, one agent creates a web page consisting of a graph of intraday stock prices notated by the temporal location of each new news story about that stock. This agent advertises a database whose records consist of a ticker symbol and a URL (of the corresponding page). Only after some agent requests the URL for a new stock does this agent actually begin the process of collecting the different types of information (from other agents) needed to produce the desired graphics.

Currently, the main WARREN system consists of 6 simple information agents (two different stock tickers, news, two SEC Edgar agents for both historical fundamental data and textual annual report (10-K) retrieval, and the matchmaker) 2 simple integration agents (one for price and news graphs, one for the NAIC "stock check list" analysis of historical sales, EPS, and pricing data) and a portfolio interface agent that interacts with the user using a web browser and forms. The portfolio agent can be customized to present different types of users different kinds of information. Agents can enter and exit the system at any time, in any order, and with any name, organized as an unstructured team. An experimental study of the various tradeoffs of the alternate organizations discussed here is underway, beginning with matchmade and brokered systems.

Acknowledgments

This work has been supported in part by ARPA contract F33615–93–1–1330, in part by ONR contract N00014–95–1–1092, and in part by NSF contract IRI–9508191.

References

Chu-Carroll, J., and Carberry, S. 1995. Communication for conflict resolution in multi-agent colaborative planning. In *Proceedings of the First International Conference on Multi-Agent Systems*, 49–56. San Francisco: AAAI Press.

Cohen, P. R., and Levesque, H. J. 1990. Intention is choice with commitment. *Artificial Intelligence* 42(3):213–261.

Cohen, P., and Levesque, H. 1995. Communicative actions for artificial agents. In *Proceedings of the First International Conference on Multi-Agent Systems*, 65–72. San Francisco: AAAI Press.

Collet, C.; Huhns, M.; and Shen, W. 1991. Resource integration using a large knowledge base in Carnot. *Computer*.

Corkill, D. D.; Lesser, V. R.; and Hudlická, E. 1982. Unifying data-directed and goal-directed control: An example and experiments. In *Proceedings of the National Conference on Artificial Intelligence*, 143–147.

Decker, K. S., and Lesser, V. R. 1993. Quantitative modeling of complex computational task environments. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 217–224.

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

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

Finin, T.; Fritzson, R.; McKay, D.; and McEntire, R. 1994. KQML as an agent communication language. In *Proceedings of the Third International Conference on Information and Knowledge Management CIKM'94*. ACM Press.

Garvey, A.; Humphrey, M.; and Lesser, V. 1993. Task interdependencies in design-to-time real-time scheduling. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 580–585.

Genesereth, M., and Katchpel, S. 1994. Software agents. *Communications of the ACM* 37(7):48–53,147.

Gmytrasiewicz, P. J.; Durfee, E. H.; and Wehe, D. K. 1991. A decision-theoretic approach to coordinating multiagent interactions. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 62–68.

Gruber, T. 1993. Toward principles for the design of ontologies used for knowledge sharing. Technical Report KSL-93-4, Knowledge Systems Laboratory, Stanford University.

Jennings, N. R. 1993. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review* 8(3):223–250.

Oates, T.; Prasad, M. V. N.; Lesser, V. R.; and Decker, K. S. 1995. A distributed problem solving approach to cooperative information gathering. In *AAAI Spring Symposium on Information Gathering in Distributed Environments*.

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

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

Sycara, K., and Zeng, D. 1995. Task-based multi-agent coordination for information gathering. In *AAAI Spring Symposium on Information Gathering in Distributed Environments*.

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

Tambe, M. 1995. Recursive agent and agent group tracking in a real-time, dynamic environment. In *Proceedings of the First International Conference on Multi-Agent Systems*, 368–375. San Francisco: AAAI Press.

Walker, M. 1994. Planning to converse with an inference-limited agent. In *AAAI-94 Workhop on Planning for Interagent Communication*, 53–62.

Wellman, M. 1993. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research* 1:1–23.

Wiederhold, G.; Wegner, P.; and Cefi, S. 1992. Toward megaprogramming. *Communications of the ACM* 33(11):89–99.

Williamson, M.; Decker, K.; and Sycara, K. 1996. Unified information and control flow in hierarchical task networks. In *Proceedings of the AAAI-96 workshop on Theories of Planning, Action, and Control*.