Electric Elves: Applying agent technology to support human organizations

Hans Chalupsky, Yolanda Gil, Craig A. Knoblock, Kristina Lerman,
Jean Oh, David V. Pynadath, Thomas A. Russ, Milind Tambe
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292

Abstract

The operation of a human organization requires dozens of everyday tasks to ensure coherence in organizational activities, to monitor the status of such activities, to gather information relevant to the organization, to keep everyone in the organization informed, etc. Teams of software agents can aid humans in accomplishing these tasks, facilitating the organization's coherent functioning and rapid response to crises, while reducing the burden on humans. Based on this vision, this paper reports on *Electric Elves*, a system that has been operational, 24/7, at our research institute since June 1, 2000.

Tied to individual user workstations, fax machines, voice, mobile devices such as cell phones and palm pilots, Electric Elves has assisted us in routine tasks, such as rescheduling meetings, selecting presenters for research meetings, tracking people's locations, organizing lunch meetings, etc. We discuss the underlying AI technologies that led to the success of Electric Elves, including technologies devoted to agent-human interactions, agent coordination, accessing multiple heterogeneous information sources, dynamic assignment of organizational tasks, and deriving information about organization members. We also report the results of deploying Electric Elves in our own research organization.

Introduction

The operation of a human organization involves dozens of critical everyday tasks to ensure coherence in organizational activities, to monitor the status of such activities, to obtain information relevant to the organization, to keep everyone in the organization informed, etc. These activities are often well-suited for software agents, which can devote significant resources to perform these tasks, thus reducing the burden on humans. Indeed, teams of such software agents, which include proxy agents that act on behalf of humans, would enable organizations to act coherently, to attain their mission goals robustly, to react to crises swiftly, and to adapt to events dynamically. Such agent teams could assist all organizations, including the military, civilian disaster response organizations, corporations, and universities and research institutions.

Within an organization, we envision agents assisting in *all* of its day-to-day functioning. For a research institution, agents may facilitate activities such as meeting (re)scheduling, selecting presenters for research meetings, composing papers, developing software and deploying people and equipment for out-of-town demonstrations. For a disaster response organization, agents may facilitate the teaming of people and equipment to rapidly respond to crises (e.g., earthquakes), to monitor the progress of any such for rapid response, etc. To accomplish such goals, each person in an organization will have an agent proxy. For instance, if an organizational crisis requires an urgent deployment of a team of people and equipment, then agent proxies could dynamically volunteer for team membership on behalf of the people or resources they represent, while also ensuring that the selected team collectively possesses sufficient resources and capabilities. The proxies must also manage efficient transportation of such resources, the monitoring of the progress of individual participants and of the mission as a whole, and the execution of corrective actions when goals appear to be endangered.

Based on the above vision, we have developed a system called *Electric Elves* that applies agent technology in service of the day-to-day activities of the Intelligent Systems Division of USC/ISI. Electric Elves is a system of about 15 agents, including nine proxies for nine people, plus two different matchmakers, one flight tracker and one scheduler running continuously for past several months. This paper discusses the tasks performed by the system, the research challenges it faced and its use of AI technology in overcoming those challenges.

One key contribution of this paper is understanding the challenges faced in deploying agents to support organizations. In particular, the complexity inherent in human organizations complicates all of the tasks agents must perform. First, since agents must interact with humans, issues of *adjustable autonomy* become critical. In particular, agents acting as proxies for people must automatically adjust their own autonomy, e.g., avoiding critical errors, possibly by letting people make important decisions while autonomously making the ore routine decisions. Second, to accomplish their goals, agents must be provided reliable access to information. Third, people have a wide variety of capabilities, interests, preferences and engage in many different tasks. To enable teaming among such people for crisis response or other organizational tasks, agents acting as their proxies must represent and reason with such capabilities and interests. We thus require powerful matchmaking capabilities to match both interests and capabilities. Fourth, coordination of all of these different agents, including proxies, is itself a significant research challenge. Finally, the entire agent system must scale-up: (i) it must scale-up to in the sense of running continually 24 hours a day 7 days a week (24/7) for months at a time; (ii) it must scale-up in the number of agents to support large-scale human organizations.

The Electric Elves

In the Electric Elves project we have developed technology and tools for deploying agents into human organizations to help with organizational tasks. We describe the application of the Electric Elves to two classes of tasks. First, we describe the problem of coordinating activities within an individual research project. These tasks must be tightly coordinated and a significant amount of information is known in advance about the participants and their goals and capabilities. Second, in order to demonstrate the capabilities of the system in a more open environment, we applied the system to the problem of meeting planning with participants outside the organization where some of the necessary information about participants is not known in advance.

Coordinating Project Activities

We have currently deployed our agents to help coordinate the everyday activities of a research project. The task of the agents is to keep the project running smoothly, rescheduling meetings when someone is delayed, ordering food for meetings or if someone has to work late, and identifying speakers for research meetings. Each person in the project is assigned their own personal proxy agent, which represents that person to the agent system.

Figure 1: PDA (Palm VII) with GPS device

A proxy agent keeps track of a project member's current location using several different information sources, including their calendar, Global Position System (GPS) device when outside of the building (Fig. 1), infrared communications within the building, and computer activity. When a proxy agent notices that someone is not attending a scheduled meeting or that they are located too far away to make it to a scheduled meeting in time, then their agent sends them a request using a wireless device (i.e., a cell phone or Palm Pilot) asking if they want to cancel the meeting, delay the meeting, or have the meeting proceed without them. If a user responds, their decision is then communicated to the other participants of the scheduled meeting. If they are unable to respond, the agent must make a decision autonomously.

For weekly project meetings, the agents coordinate the selection of the presenter and arrange food for the meetings. Once a week an auction is held where all of the meeting participants are asked about their capability and willingness to present at the next meeting. Then the system compiles the bids, selects a presenter, and notifies all of the attendees who will be presenting at the next project meeting. The agents also arrange food for lunch meetings. They order from a set of nearby restaurants, select meals that were highly rated by others, and fax the orders directly to the restaurant with instructions for delivery.

In cases when a visitor is coming to visit a project, the agents monitor the flight that the person is arriving on. If a visitor is arriving the same day of the meeting, the agent tracks their flight status and send a notification once the visitor arrives. If they are late, then the agents will delay the scheduled meetings. If they are arriving the day before a visit, then the agents will send a welcome fax to the hotel where the visitor is staying with a list of highly-rated restaurants nearby.

Some of the technical challenges in building this application are in determining how much autonomy the agents should assume on behalf of the user, dynamically building agent teams, determining how to assign the organizational tasks (e.g., presentations), and providing access to online data such as calendars, flight information, and restaurants.

Organizing External Meetings

To demonstrate how the technology supports less structured environments, we also applied the Electric Elves to the task of planning and coordinating ad hoc meetings at conferences and workshops involving individuals across different organizations. The system identifies people that have similar research interests, coordinates scheduling a meeting with those people, locates a suitable restaurant for a meeting that takes into account dietary constraints, and makes a reservation using an online reservation service.

In order to identify individuals that have related interests, the agents use an online bibliography service that provides a list of all of the papers written by an individual. When a person is going to a meeting, their agent can check an online source to locate individuals that are going to the same meeting and then build a model of the research interests of the different participants based on their publications. Given this information, the agent presents to the user a list of people that they might want to meet with along with their matching areas of interest.

Once the set of participants have been selected, the user proposes a time for the meeting and the agent sends out an invitation to each of the potential attendees. If they are part of the Electric Elves network, the invitation is sent to their own agent. Otherwise the invitation is sent via fax or email with instructions on how to confirm (or decline) the meeting.

Once the agent has finalized the set of participants for a meeting, it selects an appropriate place to have the meeting. It does this by checking for any known dietary restrictions and uses that information to identify suitable cuisine types.

Next, the agent goes out to an online restaurant reservation site to find the set of restaurants closest to the given location and matches up these restaurants with a restaurant review site to select the high-quality restaurants. The user selects from a small set of close, highly-recommended restaurants and the agent then makes a reservation for the meeting using the online reservation system.

This application highlights two additional technical challenges: gathering information about people from other organizations and ensuring the robustness of the interaction with online sources that change frequently.

Underlying Technologies

In this section we describe how we addressed some of the technical challenges, namely the issues of interacting with human users within an organization, providing reliable access to organization-related data, dynamic assignment of organizational tasks, deriving knowledge about the participants in an organization, and coordination of agent teams.

Agent Interactions with Human Users

Agents in the Electric Elves domain must often take actions on behalf of the human users. Specifically, a user's agent proxy (named "Friday" after Robinson Crusoe's servant and companion) can take autonomous actions to coordinate collaborative activities (e.g., meetings). Friday's decision making on behalf of a person naturally leads to the issue of *adjustable autonomy*. An agent has the option of acting with full autonomy (e.g., delaying a meeting, volunteering the user to give a presentation, ordering a meal for the user). On the other hand, it may act with reduced or no autonomy, instead transferring decision-making control to a person (e.g., asking its user what to do). Clearly, the more decisions that Friday makes autonomously, the more time and effort it saves its user. Yet, given the high uncertainty in Friday's knowledge of its user's state and preferences, it could potentially make very costly mistakes while acting autonomously. For example, it may order an expensive dinner when the user is not hungry, or volunteer a busy user to give a presentation. Thus, each Friday must make intelligent decisions about when to consult its user and when to act autonomously.

Our initial attempt at adjustable autonomy was inspired by CAP (Mitchell *et al.* 1994), an agent system for advising a user on scheduling meetings. As with CAP, each Friday tried to learn its user preferences using decision trees under C4.5 (Quinlan 1993). Unfortunately, one problem became apparent when applying this technique in Electric Elves: a user would not grant autonomy to Friday in making certain decisions (e.g., do not volunteer the user to give a presentation without asking), but s/he would sometimes be unavailable to provide any input at decision time. Thus, a Friday would end up waiting indefinitely for user input and miscoordinate with its teammates. To address this problem, we modified the system so that if a user did not respond within a fixed time limit, Friday acted autonomously according to its learned decision tree. Unfortunately, when we deployed the system in our research group, it led to some dramatic failures. For instance, one user's proxy automatically volunteered him to give a presentation, even though he was unwilling to do so. C4.5 had over generalized from a few examples to create an incorrect rule. Although Friday tried asking the user at first, because of the timeout, it had to eventually follow the incorrect rule and take the undesirable autonomous action.

It was clear, based on this experience, that the team context in Electric Elves would cause difficulties for existing adjustable-autonomy techniques (Dorais *et al.* 1998; Ferguson, Allen, & Miller 1996; Mitchell *et al.* 1994) that focused on solely individual human-agent interactions. Therefore, we developed a novel, decision-theoretic planning approach that used Markov Decision Processes (MDPs) (Puterman 1994) to support explicit reasoning about team coordination. The MDPs used in our framework (Scerri, Pynadath, & Tambe 2001) provide Friday with a novel three-step approach to adjustable autonomy: (i) Before transferring decision-making control, an agent explicitly weighs the cost of waiting for user input and any potential team miscoordination against the likelihood and cost of erroneous autonomous action; (ii) When transferring control, an agent does not rigidly commit to this decision (as is often the case in previous work), but it instead flexibly reevaluates when its user does not respond, sometimes reversing its decision and taking back autonomy to ensure team coordination; (iii) Rather than force a risky decision in situations requiring autonomous action, an agent changes its coordination arrangements by postponing or reordering activities to potentially buy time to lower decision cost/uncertainty. Since these coordination decisions and actions incur varying costs and benefits over time, agents look ahead over the different sequences of possible changes in coordination and plan a policy of action that maximizes team welfare.

The agent follows the first step of our approach through team-related components within its MDP model of the costs and benefits of its available actions. Thus, the MDP's decision-theoretic selection of optimal policies trades off individual preferences against the team's needs. The policies generated from the MDP support the second step of our approach by providing the necessary flexibility and responsiveness in autonomy decisions. The agent can immediately respond to any change of state by following the policy's specified action for the new state. In this respect, the agent's decision making is an ongoing process rather than a single decision, as the agent acts according to its MDP policy throughout the entire sequence of states it finds itself in. We achieve the third step of our approach by having each agent consider the different costs, both present and future, of team miscoordination vs. erroneous actions. In the meeting scenario, changes in coordination are essentially delaying actions. Such changes in coordination could, among other things, buy time to reduce the uncertainty or cost. MDPs are especially suitable for producing such a plan because they generate policies while looking ahead at all of the possible outcomes.

We have designed and implemented MDPs that model Friday's decisions on meeting rescheduling, volunteering its user to give a presentation, and selecting *which* user should give a presentation. For instance, consider one possible policy, generated from an MDP for the rescheduling of meetings. If the user has not arrived at the meeting five minutes prior to its scheduled start, this policy specifies "ask the user what to do" and then "wait". If the user does not arrive by the time of the meeting, the policy again specifies "wait", so the agent continues acting without autonomy. However, if the user *still* has not arrived five minutes after the meeting is scheduled to start, then the policy chooses "delay by 15 minutes", which the agent then executes autonomously. In the future, we plan to apply our MDP-based framework to other decisions (currently performed without any autonomy) within Electric Elves, such as ordering meals, accepting meeting invitations, and selecting restaurants. In addition, the current MDP framework supports some learning of likelihoods (e.g., the probability that the user will arrive to the meeting on time), but we are planning to extend the role of learning to allow further personalization.

Flexible Assignment of Tasks

The human agents and software agents in our organization perform a wide variety of tasks that are often interrelated. Agents often need to delegate a subtask to another agent capable of performing it (e.g., reserve a meeting room), invoke another agent to gather and report back necessary information (e.g., find the location of a person), or rely on another agent to execute some task in the real world (e.g., attend a lunch meeting). Simple agent matchmaking is sufficient in many multi-agent systems where agents perform one (or at most a few) kind of task, and their capabilities are designed by the system developers to fit the interactions anticipated among the agents. In contrast, our agents are complex and heterogeneous, and the agents that issue a request cannot be expected to be aware about what other agents are available and how they are invoked.

We have developed an agent matchmaker called PHOSPHORUS, which draws from previous research on matching problem solving goals and methods within the EXPECT architecture (w. R. Swartout & Gil 1995; Gil & Gonzalez 1996). The main features of this approach are: 1) a declarative language to express task descriptions that includes rich parameter type expressions to qualify task types; 2) task descriptions are fully translated into description logic to determine subsumption relations among tasks; 3) task descriptions are expressed in terms of domain ontologies, which provide a basis for relating and reasoning about different tasks and enables reformulation of tasks into subtasks. A more detailed description of PHOSPHORUS can be found in (Gil & Ramachandran 2001).

Agent capabilities and requests are represented as verb clauses with typed arguments (as in a case grammar), where each argument has a name (usually a preposition) and a parameter. The type of a parameter may be a specific instance, an abstract concept (marked with `spec-of`), an instance type (marked with `inst-of`), and extensional or intensional sets of those three types. Here are some examples of capabilities of some researchers and project assistants:

```
"agents that can discuss Phosphorus"
   ((capability (discuss (obj Phosphorus-project)))
    (agents (gil surya chalupsky russ)))

"agents that can setup an LCD projector in a meeting room"
   ((capability (setup (obj (?v is (inst-of lcd-projector)))
                       (in (?r is (inst-of meeting-room)))))
    (agents (itice)))
```

Requests are formulated in the same language, and can ask about general types of instances (e.g., what agents can setup any kind of equipment for giving research presentations in a meeting room).

Description logic and subsumption reasoning are used to relate different task descriptions. Both requests and agent capabilities are fully translated into Loom (MacGregor 1991). Loom's classifier is now able to reason about these definitions and places them in a lattice, where more general definitions subsume more specific ones. Notice that this subsumption reasoning uses the definitions of the domain terms and ontologies. As a result, the capability to "setup equipment" will subsume one to "setup LCD projector", because according to the domain ontologies equipment subsumes LCD projector. The capabilities are automatically organized according to their definitions, and they can be compared based on their place in the lattice.

PHOSPHORUS performs *task reformulations* when there are no agents with capabilities that subsume a request. In that case, it may be possible to fulfill the request by decomposing it into subtasks. This allows a more flexible matching than is possible if one required an exact match for capabilities and requests. PHOSPHORUS supports set reformulations, which break down a task on a set into its individual elements, and covering reformulations, which decompose a task based on the disjoint partitions or subclasses of its arguments. For example, in order to setup a meeting to discuss the Electric Elves project at ISI the meeting organizer would issue a request to the matcher. This request cannot be fullfilled by a single agent since no single researcher is involved in all the aspects of the Electric Elves project. PHOSPHORUS returns a decomposition of this request based on the subprojects:

```
(COVERING -name ARIADNE-PROJECT
               -matches KNOBLOCK MINTON LERMAN
          -name PHOSPHORUS-PROJECT
               -matches GIL SURYA CHALUPSKY RUSS
          -name TEAMCORE-PROJECT
               -matches
                 (COVERING
                 -name ADJUSTABLE-AUTONOMY-PROJECT
                      -matches TAMBE SCERRI PYNADATH
                 -name TEAMWORK-PROJECT
                      -matches TAMBE PYNADATH MODI)
          -name ROSETTA-PROJECT
               -matches GIL CHALUPSKY)
```

The flexibility of our approach is illustrated in this example in that the requesting agent did not need to be aware of the details of the Electric Elves project, and gets from the reply subsets of agents that are able to fulfill complementary parts of the request.

The SHADE matchmaker (Kuokka & Harada 1995) also matched agent capabilities using logic descriptions, but the basic matching operation was done by unification and did not exploit domain ontologies to relate different terms. In RETSINA (Sycara *et al.* 1999), agent matchmaking is also done using description logic but only to match the parameters of the capability descriptions, while PHOSPHORUS translates the entire expression for a more thorough match. On the other hand, RETSINA has been used with very large communities of agents, and incorporates information retrieval techniques.

Many additional challenges lay ahead regarding capability representations for people within the organization. For example, in principle anyone has the capability to call a taxi for a visitor (and will do so if necessary), but project assistants are the preferred option. Depending on upcoming deadlines, a researcher may be capable but not willing to participate in a visitor's schedule. Future extensions to the language will also be needed to express additional properties of agents, such as reliability, efficiency, and invocation guidelines.

Reliable Access to Information

Timely access to up-to-date information is crucial to the successful planning and execution of tasks in the Electric Elves organization. Agents making decisions on behalf of human users need to extract information from multiple heterogeneous information sources, which include internal organizational databases (personal schedules, staff lists), and external Web sites such as airline schedules, restaurant information, traffic and weather updates, etc. For example, in order to pick a restaurant for a scheduled lunch meeting, the agents access the Restaurant Row Web site to get the locations of restaurants that meet the specified criteria, e.g., dietary restrictions. Wrappers extract data from information sources and make it available to other applications, such as the Electric Elves agents. Moreover, wrappers enable the sources, including Web sites, to be queried as if they were databases. A critical part of the wrapper is a set of extraction rules, often based on "landmarks" or sequences of tokens, that enable the wrapper to quickly locate the beginning and end of data to be extracted from a page returned by the Web source in response to some query.

Within the Electric Elves project, we use the Ariadne system (Knoblock *et al.* ; 1998) to learn wrappers from example pages in which relevant data has been labeled by the user. Previous research has primarily focused on applying machine learning techniques to rapidly generate wrappers (Muslea, Minton, & Knoblock 2001; Hsu & Dung 1998; Kushmerick 2000). Few attempts were made to provide the capability to validate data, detect failures (Kushmerick 1999) and repair wrappers when the underlying sources change in a way that breaks the wrapper. The ability to automatically monitor external information sources and repair wrappers when errors are detected is a critical component of a robust dynamic organization.

We address the problem of wrapper verification by applying machine learning techniques to learn a set of patterns that describe the information being extracted for each data field. Since the information for a single field can vary considerably, the system learns a statistical distribution of patterns. Wrappers can be verified by comparing newly extracted data to the learned patterns. When a significant difference is found, an operator can be notified or we can automatically launch the wrapper repair process.

The learned patterns represent the structure, or format, of data as a sequence of words and wildcards. Wildcards represent syntactic categories to which words belong — alphabetic, numeric, capitalized, *etc.* — and allow for multilevel generalization. For example, a set of street addresses all start with a pattern "*_Number_ Capitalized_*": a numeric token followed by a capitalized word. The algorithm we developed (Lerman & Minton 2000) finds all statistically significant starting and ending patterns in a set of positive examples of the data field. A pattern is said to be significant if it occurs more frequently than would be expected by chance if the tokens were generated randomly and independently of one another. Our approach is similar to work on grammar induction (Carrasco & Oncina 1994; Goan, Benson, & Etzioni 1996), but our pattern language is better suited to capturing the regularities in small data fields (as opposed to languages). For the verification task, we learn the patterns from data extracted by the wrapper that is known to be correct (training examples). In the verification phase, the wrapper generates a test set of examples from pages retrieved using the same or similar set of queries. If the patterns describe statistically the same (at a given significance level) proportion of the test examples as the training examples, the wrapper is judged to be correct; otherwise, it is judged to have failed.

The most common causes of wrapper failure are changes in Web site format, even minor reorganizations of a page, that break the wrapper's data extraction rules. However, since the content of the fields tends to remain the same, it is often possible to automatically repair the wrapper by learning new extraction rules. For this purpose, we exploit the patterns learned for verifying data to locate correct examples of data on new pages. For example, while the wrapper for Restaurant Row was working correctly, we managed to acquire several examples of restaurant addresses, and the verification algorithm learned that some fraction of the examples start with the pattern "*_Number_ Capitalized_*" and end with the pattern "Avenue". If the Restaurant Row changes its format to look more like the Zagat Web site, the wrapper will no longer extract correct addresses. The verification algorithm will detect the failure because extracted data will not be described by the learned patterns. However, since restaurant addresses will still start with the pattern "*_Number_ Capitalized_*" and end with the pattern "Avenue", we should be able to identify addresses on the changed pages. Once the required information has been located, these examples, along with the new pages, are provided to the wrapper generation system to learn data extraction rules for the changed site. In order to identify the correct examples of data on the changed pages we leverage both the prior knowledge about the content of data, as captured by the learned patterns, and *a priori* expectations about data. We can reasonably expect the same data field to appear in roughly the same position and in a similar context on each page; moreover, we expect at least some of the data to remain unchanged. Of course, the latter assumption is violated by information that changes on a regular basis, such as traffic and flight arrival information, though we may still rely on patterns to identify correct examples.

Our approach can be extended to automatically create wrappers for new information sources using data extracted from a known source. Thus, once we learn what restaurant addresses look like, we can use this information to extract addresses from any other yellow pages-type source, and use the extracted data to create a wrapper for the new source. Such cross-site wrapper creation, as well automatic extraction of data from more complex sources, such as lists and tables, is a focus of our current research.

Deriving Organizational Knowledge from Unstructured Sources

As mentioned above, an agent-assisted organization crucially depends on access to accurate and up-to-date information about the humans it supports as well as the environment they operate in. While some of this information can be provided directly from existing databases and online sources, other information such as people's expertise, capabilities, interests, etc. will often not be available explicitly and might need to be modeled by hand. In a dynamic environment such as Electric Elves, however, manual modeling is only feasible for information that is relatively static. For example, if at some conference we want to select potential candidates for a lunch meeting with Yolanda Gil based on mutual research interests, it is not feasible to manually model relevant knowledge about each person on the conference roster before such a selection can be made.

To support team-building tasks such as inviting people for a lunch meeting, finding people potentially interested in a presentation or research meeting, finding candidates to meet with a visitor, etc., we developed a matchmaking service called the Interest Matcher. It can match people based on their research interests but also take other information into account such as involvement in research projects, present and past affiliation, universities attended, etc. To minimize the need for manual modeling in a dynamic environment, we integrated statistical match techniques from the area of information retrieval (IR) with logic-based matching performed by a knowledge representation (KR) system. The IR techniques work well with unstructured text sources available online on the Web, which is the form information is typically available in organizations, while the KR system facilitates declarative modeling of the decision process, modeling of missing information, explanation and also customization.

The matchmaker is built on top of the PowerLoom KR system which is the successor to Loom (MacGregor 1991). PowerLoom uses a variant of KIF (Genesereth 1991) as its input language, and its inference, explanation and partial-match capabilities are important to support the matchmaking task. The matchmaker's knowledge base contains an ontology of research topic areas and associated relations, rules formalizing the matchmaking process, as well as various manually modeled, relatively static information about staff members, research projects, etc. To perform a particular matchmaking task, a requesting agent sends a message containing an appropriate PowerLoom query to the Interest Matcher. For example, to find candidates for lunch with Yolanda Gil, the following query could be used:

```
(retrieve all ?x (should-meet ?x Gil))
```

The should-meet relation and one of its supporting relations are defined as follows in PowerLoom:

```
(defrelation should-meet ((?p1 Person) (?p2 Person))
  :⇐ (or (interests-overlap ?p1 ?p2)
         (institution-in-common ?p1 ?p2)
         (school-in-common ?p1 ?p2)))

(defrelation interests-overlap ((?p1 Person) (?p2 Person))
  :⇐ (exists (?interest1 ?interest2)
         (and (research-interest ?p1 ?interest1)
              (research-interest ?p2 ?interest2)
              (or (subset-of ?interest1 ?interest2)
                  (subset-of ?interest2 ?interest1)))))
```

should-meet is defined by combining three more basic relations which provide reasons why two people might want to meet. For more specific purposes, any of the more basic relations such as interests-overlap could be queried directly by a client, which is one of the advantages gained from using a general purpose KR system as the matching engine. Note, that for interests-overlap we only require a subsumption relationship, e.g., interest in planning would subsume (or overlap with) interest in hierarchical planning.

To answer the query above, the matchmaker generates the set of people in its knowledge base satisfying the should-meet relation and returns it as a result (usually, the candidate set is further constrained and does not include everybody in the KB). In an ideal world, the matchmaking KB would be complete. In the real world, this will usually not be the case, particularly when the organization interacts with outsiders such as conference attendees. To deal with this incompleteness, we allow a requesting agent to introduce new individuals into the KB and automatically infer limited structured knowledge - their research interests - about them by analyzing relevant unstructured text sources on the Web.

The key idea to this step is that people's research interests are implicitly documented in their publication record. To make these interests explicit, we first associate each research topic in the PowerLoom topic ontology with a statistical representation comprised of a set of abstracts of research papers representative of the topic. These topic sets are determined automatically by querying a bibliography search engine such as Cora or the NEC ResearchIndex with seed phrases representative of the topic (access to such Web sources is facilitated by Ariadne wrappers). We then query the same search engine for publication abstracts of a particular researcher and then classify them by computing statistical similarity measures between the researcher's publications and the topic sets determined before. When the similarity surpasses an empirically determined threshold, an appropriate interest assertion is added to the matchmaker KB that can then be exploited in the matchmaking process described above.

We use a standard IR vector space model to represent document abstracts and compute similarity by a cosine measure and by weighting terms based on how well they signify particular topic classes (Salton & McGill 1983). We also use our own aggressive stemmer to reduce the number of features that need to be considered for similarity computations.

The dynamic derivation of interests from unstructured online sources sets our approach apart from the one described in (Sycara & Zeng 1996) that solely relies on manually specified interests. Extending KR with IR can also increase robustness for the case of an incomplete topic ontology, since we can statistically match researchers' publications directly. Conversely, IR matching can benefit from KR matching for the case where two researchers do not have similar publication records but can be related via similarity to a common or hierarchically related topic. The smooth combination of statistical and logical reasoning is a non-trivial problem, however, and still provides room for further research and improvement.

Coordination of Component Agents

The various agents and software components described in this section are autonomous, heterogeneous, and distributed over a variety of platforms and research groups. Yet, these diverse agents must work together to accomplish the complex tasks required by Electric Elves. For instance, to plan a meeting, the interest matcher must first find a list of potential attendees, the Friday of each potential attendee must decide whether s/he will attend or not, the capability matcher must identify any dietary restrictions of the confirmed attendees, and the reservation site wrapper must identify possible restaurants and make the final reservation. In addition to the low-level communication issues raised by such a task, there is the more complicated problem of getting these heterogeneous agents to work together as a team. In other words, each of these agents must execute its part in coordination with the others, so that it executes its tasks at the correct time and passes on the results to the agents who need them.

However, constructing teams of such agents remains a difficult challenge. In particular, current approaches to designing agent teams lack the general-purpose teamwork models that would enable agents to autonomously reason about the communication and coordination required in teamwork. The absence of such teamwork models makes team construction highly labor-intensive. In particular, to enable agents to autonomously reason about coordination, human developers must provide them with a large number of problem-specific coordination and communication plans. These problem-specific plans are not reusable, so we must develop new ones for each new problem. Furthermore, the resulting teams often suffer from a lack of robustness and flexibility. In a real-world domain like Electric Elves, teams face a variety of uncertainties, such as a member agent's unanticipated failure in fulfilling responsibilities (e.g., presenter is delayed in attending a meeting), members' divergent beliefs about their environment, and unexpectedly noisy or faulty communication. Without a teamwork model, it is difficult to anticipate and pre-plan for the innumerable coordination failures possible.

In Electric Elves, the agents coordinate using Teamcore, a domain-independent, decentralized, teamwork-based integration architecture (Pynadath *et al.* 1999). Teamcore integrates a general-purpose teamwork model (called STEAM (Tambe 1997)) and provides core teamwork capabilities to agents by wrapping them with Teamcore proxies (not to be confused with the Friday agents that act as *user* proxies). By interfacing with Teamcore proxies, existing agents become *team-ready* and thus able to rapidly assemble themselves into a team to solve a given problem. To this end, the Teamcore proxies form a distributed team-readiness layer for augmenting existing agents with the following social capabilities: (i) coherent commitment and termination of joint goals, (ii) team reorganization in response to member failure, (iii) selective communication, (iv) incorporation of heterogeneous agents, and (v) automatic generation of tasking and monitoring requests. Although other agent integration architectures such as OAA (Martin, Cheyer, & Moran 1999) and RETSINA (Sycara *et al.* 1996) provide capability (iv), Teamcore's use of an explicit, domain-independent teamwork model allows it to support these other necessary social capabilities as well.

Each and every agent in the Electric Elves organization (Fridays, matchers, wrappers, etc.) has an associated Teamcore proxy that records its membership in various teams and keeps track of active commitments made to these teams. Given an abstract specification of the organization and possible commitments, the Teamcore proxies *automatically* execute the necessary coordination tasks. They form joint commitments to team plans such as holding meetings, hosting and meeting with visitors, arranging lunch, etc. Teamcore proxies also communicate amongst themselves to ensure coherent execution of team plans and robust achievement of joint goals. Given their teamwork knowledge, the Teamcore proxies automatically substitute for missing roles (e.g., if the presenter is absent from the meeting) and inform each other of critical factors affecting a team plan. Finally, they communicate with their corresponding agents to monitor the agents' ability to fulfill commitments (e.g., asking Friday to monitor its user's attendance of a meeting) and to inform the agents of changes to those commitments (e.g., notifying Friday of a meeting rescheduling).

Figure 2 shows a small portion of the organization hierarchy used to model the software agents and human users, as well as the teams they belong to. It also shows a small portion of the plan hierarchy used to model possible commitments that these teams can undertake. Both figures are partial screenshots of a graphical tool that allows a human user to modify the organization and plans, thus supporting rapid incorporation of new members, teams, and tasks. Currently, there are 21 classes of such commitments for performing the Electric Elves' tasks, and there are 23 teams and subteams, covering a total of 50 individual agent roles (with some agents filling multiple roles).

Electric Elves Architecture

Electric Elves is a complex and highly heterogeneous system spanning a wide variety of component technologies and languages, communication protocols as well as operating system platforms. Figure 3 gives an overview of all the different components participating in the current version of

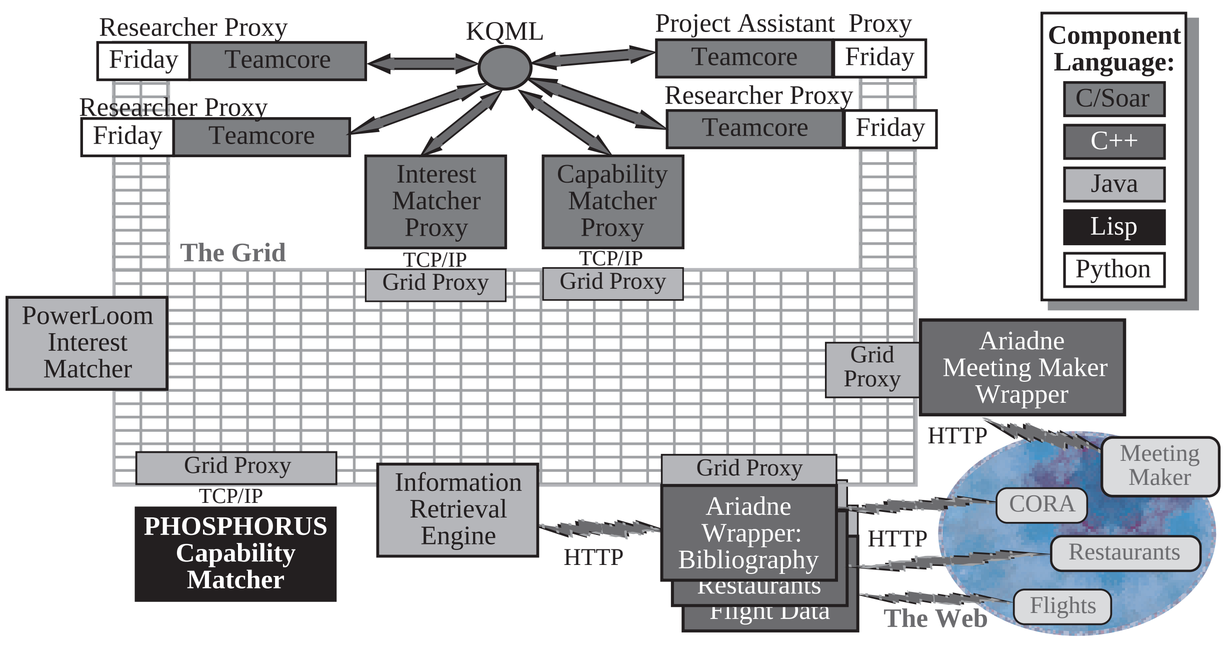
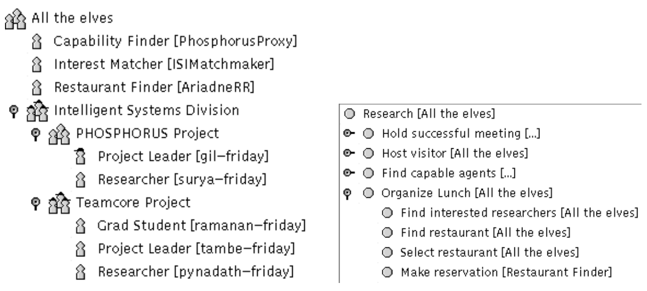


Figure 3: Electric Elves System Architecture


All the elves
Capability Finder [PhosphorusProxy]
Interest Matcher [ISIMatchmaker]
Restaurant Finder [AriadneRR]
Intelligent Systems Division
PHOSPHORUS Project
Project Leader [gil-friday]
Researcher [surya-friday]
Teamcore Project
Grad Student [ramanan-friday]
Project Leader [tambe-friday]
Researcher [pynadath-friday]

Research [All the elves]
Hold successful meeting [...]
Host visitor [All the elves]
Find capable agents [...]
Organize Lunch [All the elves]
Find interested researchers [All the elves]
Find restaurant [All the elves]
Select restaurant [All the elves]
Make reservation [Restaurant Finder]

Figure 2: Partial organization (left) and plan (right) hierarchies for the Electric Elves domain.

Electric Elves. Teamcore agents are written in Python and Soar (which is written in C), Ariadne wrappers are written in C++, the PHOSPHORUS capability matcher is written in Common-Lisp and the PowerLoom interest matcher is written in STELLA (Chalupsky & MacGregor 1999) which translates into Java. The agents are distributed across SunOS 5.7, Windows NT, Windows 2000 and Linux platforms, and use TCP/IP, HTTP and the Lockheed KQML API to handle specialized communication needs.

Tying all these different pieces together in a robust and coherent manner constitutes a significant engineering challenge. To solve this integration problem, we are using the CoABS Grid technology (Global InfoTek, Inc. 2000) developed by Global InfoTek, Inc. and ISX Corporation as part of DARPA's CoABS program. The CoABS Grid is a Java-based communication infrastructure built on top of Sun's Jini networking technology (Sun Microsystems, Inc. 1999). It provides message and service-based communication mechanisms, agent registration, lookup and discovery services, as well as message logging, security and visualization facilities. Since it is is written in Java, it runs on a wide variety of OS platforms, and it is also relatively easy to connect with non-Java technology. The special-purpose connectors between the Grid and non-Java technology are indicated by the various Grid proxy components in Figure 3.

We primarily use the CoABS Grid as a uniform transport mechanism. The actual messages sent across the Grid are in KQML format and could potentially be communicated via alternative means. Not all Electric Elves message traffic goes across the Grid. For example, the Teamcore agents communicate via their own protocol (the Lockheed KQML API) and only use the Grid to communicate with non-Teamcore agents such as the capability and interest matchers. Similarly, the information retrieval engine communicates with Ariadne wrappers directly via HTTP instead of going through the Grid.

In general, our experience with the CoABS Grid has been very positive. It has proven to be reasonably robust and up to the task of 24/7 operation (since June 2000 we have logged over 40,000 Grid messages). Overall, the Grid has provided us with basic interoperability that would have been difficult to achieve otherwise. Initially we were looking for an alternative communication solution such as using some implementation of KQML, but there was none available that satisfied all the different language, OS platform and protocol requirements. Below is an example list of synergies that resulted in part from this basic interoperability provided by the CoABS Grid:

- Simple access to Ariadne Web wrappers motivated the connection of IR with KR techniques for the purpose of the Interest Matcher.

- Access to the PHOSPHORUS capability matchmaker provides Teamcore agents with sophisticated capability reasoning that allows more accurate assembly of teams for particular tasks.
- Similarly, by using Ariadne wrappers for Meeting Maker scheduling software, flight tracking, restaurant selection, etc., Teamcore agents can access a much richer information sphere and support more complex and interesting tasks than otherwise possible.

Related Work

Several agent-based systems have been developed that support specific tasks within an organization, such as meeting scheduling (Dent *et al.* 1992) and visitor hosting (Kautz *et al.* 1994; Sycara & Zeng 1994). In contrast to these systems, we believe that our approach integrates a range of technologies that can support a variety of tasks within the organization. Agent architectures have been applied to organizational tasks (Sycara *et al.* 1996; Martin, Cheyer, & Moran 1999; Lesser *et al.* 1999), but none of them include technology for team work, adjustable autonomy, and dynamic collection of information from external sources.

To our knowledge, Electric Elves represents the first agent-based system that is used for routine tasks within a human organization. Several other areas of research have looked at complementary aspects of the problems that we aim to address. Research on architectures and systems for Computer-Supported Cooperative Work include a variety of information management and communication technologies that facilitate collaboration within human organizations (Greenberg 1991; Malone *et al.* 1997). In contrast with our work, they do not have agents associated with people that have some degree of autonomy and can make decisions on a human's behalf. Our work is also complementary and can be extended with ongoing research on ubiquitous computing and intelligent buildings (Dertouzos 1999; Lesser *et al.* 1999). These projects are embedding sensor networks and agency to control and improve our everyday physical environments. Our work could benefit from this kind of infrastructure, by making it easier to locate and contact people as well as to integrate agents that control the environment with our facilities to support organizational tasks.

Current Status

The Electric Elves system has been in use within our research group at ISI since June 1, 2000; and operating continuously 24 hours a day, 7 days a week(occasionally interrupted for bug fixes and enhancements). Usually, nine agent proxies are working for nine users, with one proxy each for a capability matcher and an interest matcher. The proxies communicate with their users using a variety of devices including our workstation display, voice, mobile phones, palm pilots; they also communicate with restaurants by sending out faxes.

Figure 4 plots the number of daily messages exchanged by the proxies for seven months (June 1, 2000 to December 31, 2000). The size of the daily counts demonstrates the large amount of coordination actions necessary in managing all of the activities such as meeting rescheduling. The high variability is due to the variance in the number of daily activities, e.g., weekends and long breaks such as the Christmas break, usually have very little activity. Furthermore, with continually increasing system stability, the amount of housekeeping activity necessary has reduced automatically.

Overall, the effectiveness of Electric Elves can be seen from several observations. First, over the past several months, few emails have been exchanged among our group members indicating to each other that they may get delayed to meetings. Instead, Friday agents automatically address such delays. Thus, the overhead of waiting for delayed members in meeting rooms has also reduced. Overall, 1128 meetings have been monitored, out of which 285 have been rescheduled, 230 automatically and 55 by hand. Thus, both autonomous rescheduling and human intervention were useful in Elves.

Furthermore, whereas in the past, one of our research group members would need to circulate emails trying to recruit a presenter for research meetings and making announcements, this overhead has almost completely vanished — weekly auctions automatically select the presenters at our research meetings. These auctions are automatically opened when the system received notification of any meeting requiring a presentation. A summary of the results is in the table 1 below. Column 1 shows the dates of the research presentations. While the auctions are held weekly, several weekly meetings over this summer were cancelled due to conference travel and vacations. Column 2 shows the total number of bids received before a decision. The key here is that auction decisions may be made with fewer than 9 bids; in fact, in one case, only 4 bids were received. The rest of the group simply did not bid until the winner was announced. Column 3 shows the winning bid. A winner typically bid <1,1>, i.e., indicating that the user it represents is both capable and willing to the presentation. Interestingly, the winner of July 27 had a bid of <0,1>, i.e., not capable but willing. Thus, the proxy team was able to settle on a winner despite the bid not being the highest possible, illustrating its flexibility. Note that the capability bids for these auctions are obtained by querying the PHOSPHORUS matchmaker. Finally, column 4 shows the results. In six of the eight times, winner was automatically selected. However, on two occasions (July 6 and Sept 19) exceptional circumstances (e.g., a visitor) required human intervention, which our proxy team could easily accommodate.

Date	# of bids	Winner<bid>	Autonomous?
July 6	7	Scerri<1,1>	No
July 20	9	Scerri<1,1>	Yes
July 27	7	Kulkarni<0,1>	Yes
August 3	8	Nair<1,1>	Yes
August 31	4	Tambe<1,1>	Yes
Sept 19	6	Visitor<-,->	No
Oct 31	7	Tambe<1,1>	Yes
Nov 21	7	Nair<1,1>	Yes

Table 1: Results for auctioning research presentation slot.

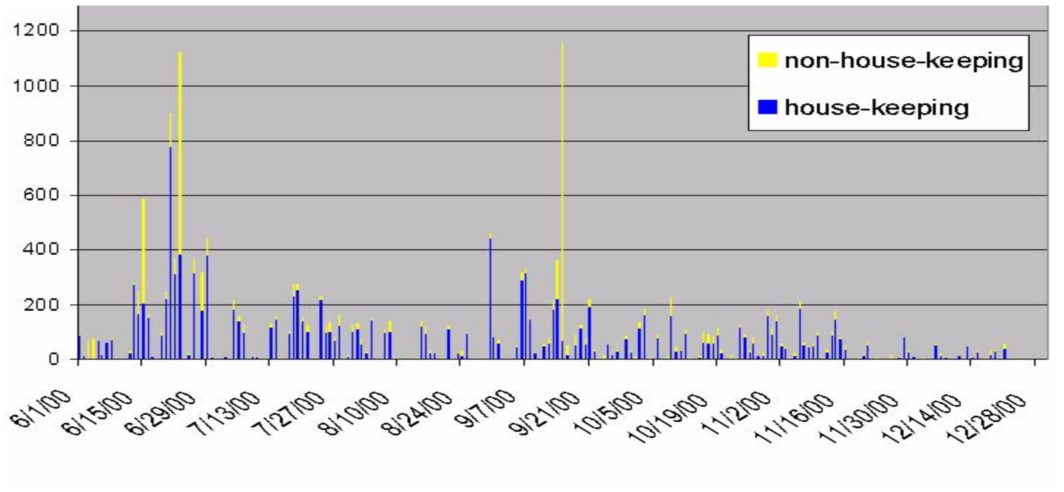


Figure 4: Number of daily coordination messages exchanged by proxies over a seven-month period.

Other benefits of Electric Elves includes a web page, where different Friday agents post their user's location, enables us to track our group members quickly; again, avoiding the overhead of trying to track them down manually. Finally, we have begun relying on our agents so heavily to order lunch that one local "Subway" restaurant owner even suggested marketing to agents: *"... more and more computers are getting to order food... so we might have to think about marketing [to them]!!"*.

Discussion

As described in this paper we have successfully deployed the Electric Elves in our own real-world organization. These agents interact directly with humans both within the organization and outside the organization communicating by email, wireless messaging, and faxes. Our agents go beyond simply automating tasks that were previously performed by humans. Because hardware and processing power is cheap, our agents can perform a level of monitoring that would be impractical for human assistants, ensuring that activities within an organization run smoothly and that events are planned and coordinated to maximize the productivity of the individuals of an organization.

In the process of building the applications described in this paper we addressed an number of key technology problems that arise in any agent-based system applied to human organizations. In particular we described how to use Markov Decision Processes to determine the appropriate degree of autonomy for the agents, how to use knowledged-based matchmaking to assign tasks within an organization, how to apply machine learning techniques to ensure robust access to the data sources, how to combine knowledge-based and statistical matchmaking techniques to derive knowledge about the participants both within and outside an organization, and how to apply multi-agent teamwork coordination to dynamically assemble teams.

There are a huge number of possible applications of this work. We plan to continue to both extend the range of applications and the underlying technologies for building the agents. One of the advantages of deploying the research in our own organization is that there is no shortage of ideas for future tasks for the Electric Elves to perform.

References

Carrasco, R., and Oncina, J. 1994. Learning stochastic regular grammars by means of a state merging method. In *Lecture Notes In Computer Science*, 862.

Chalupsky, H., and MacGregor, R. 1999. STELLA – a Lisp-like language for symbolic programming with delivery in Common Lisp, C++ and Java. In *Proceedings of the 1999 Lisp User Group Meeting*. Berkeley, CA: Franz Inc.

Dent, L.; Boticario, J.; McDermott, J.; Mitchell, T.; and Zabowski, D. 1992. A personal learning apprentice. In *Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI-92*.

Dertouzos, M. L. 1999. The future of computing. *Scientific American*.

Dorais, G. A.; Bonasso, R. P.; Kortenkamp, D.; Pell, B.; and Schreckenghost, D. 1998. Adjustable autonomy for human-centered autonomous systems on mars. In *Proceedings of the first International Conference of the Mars Society*.

Ferguson, G.; Allen, J.; and Miller, B. 1996. TRAINS-95 : towards a mixed initiative planning assistant. In *proceedings of the third conference on artificial intelligence planning systems*, 70–77.

Genesereth, M. 1991. Knowledge interchange format. In Allen, J.; Fikes, R.; and Sandewall, E., eds., *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, 599–600. San Mateo, CA, USA: Morgan Kaufmann Publishers.

Gil, Y., and Gonzalez, P. 1996. Subsumption-based matching: Bringing semantics to goals. In *International Workshop on Description Logics*.

Gil, Y., and Ramachandran, S. 2001. Phosphosrus: A task-based agent matchmaker. In *Proceedings of the Fifth International Conference on Autonomous Agents*.

Global InfoTek, Inc. 2000. The CoABS Grid. `http://coabs.globalinfotek.com/coabs_public/coabs_pdf/gridvision.pdf`.

Goan, T.; Benson, N.; and Etzioni, O. 1996. A grammar inference algorithm for the world wide web. In *Proc. of the AAAI Spring Symposium on Machine Learning in Information Access*.

Greenberg, S., ed. 1991. *Computer Supported Cooperative Work and Groupware*. London: Academic Press.

Hsu, C., and Dung, M. 1998. Generating finite-state transducers for semi-structured data extraction from the web. *Journal of Information Systems* 23(8):521–538.

Kautz, H.; Selman, B.; Coen, M.; and Ketchpel, S. 1994. An experiment in the design of software agents. In *Proceedings of the Twelfth National Conference on Artificial Intelligence, AAAI-94*.

Knoblock, C. A.; Lerman, K.; Minton, S.; and Muslea, I. Accurately and reliably extracting data from the web: A machine learning approach. *Data Engineering Bulletin*. Forthcoming.

Knoblock, C. A.; Minton, S.; Ambite, J. L.; Ashish, P. J. M. N.; Muslea, I.; Philpot, A. G.; and Tejada, S. 1998. Modeling web sources for information integration. In *Proc. Fifteenth National Conference on Artificial Intelligence*.

Kuokka, D., and Harada, L. 1995. Modeling web sources for information integration. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence (IJCAI-95)*.

Kushmerick, N. 1999. Regression testing for wrapper maintenance. In *Proc. of the 16th National Conference on Artificial Intelligence AAAI-1999*, 74–79.

Kushmerick, N. 2000. Wrapper induction: efficiency and expressiveness. *Artificial Intelligence Journal* 118(1-2):15–68.

Lerman, K., and Minton, S. 2000. Learning the common structure of data. In *Proc. of the 17th National Conference on Artificial Intelligence AAAI-2000*, 609–614.

Lesser, V.; Atighetchi, M.; Benyo, B.; Horling, B.; Raja, A.; Vincent, R.; Wagner, T.; Xuan, P.; and Zhang, S. X. 1999. The umass intelligent home project. In *Proceedings of the Third Annual Conference on Autonomous Agents (AGENTS-99)*, 291–298.

MacGregor, R. 1991. Inside the LOOM description classifier. *ACM SIGART Bulletin* 2(3):88–92.

Malone, T. W.; Crowston, K.; Lee, J.; Pentland, B.; Dellarocas, C.; Wyner, G.; Quimby, J.; Osborne, C.; and Bernstein, A. 1997. *Tools for inventing organizations: Toward a handbook of organizational processes*. Center for Coordination Science Working Paper No. 198.

Martin, D. L.; Cheyer, A. J.; and Moran, D. B. 1999. The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence* 13(1-2):92–128.

Mitchell, T.; Caruana, R.; Freitag, D.; McDermott, J.; and Zabowski, D. 1994. Experience with a learning personal assistant. *Communications of the ACM* 37(7):81–91.

Muslea, I.; Minton, S.; and Knoblock, C. 2001. Hierarchical wrapper induction for semistructured information sources. *Journal of Autonomous Agents and Multi-Agent Systems*. (to appear).

Puterman, M. L. 1994. *Markov Decision Processes*. John Wiley & Sons.

Pynadath, D. V.; Tambe, M.; Chauvat, N.; and Cavedon, L. 1999. Toward team-oriented programming. In Jennings, N. R., and Lespérance, Y., eds., *Intelligent Agents VI: Agent Theories, Architectures and Languages*. Springer-Verlag. 233–247.

Quinlan, J. R. 1993. *C4.5: Programs for machine learning*. San Mateo, CA: Morgan Kaufmann.

Salton, G., and McGill, M. 1983. *Introduction to Modern Information Retrieval*. Tokio: McGraw-Hill.

Scerri, P.; Pynadath, D. V.; and Tambe, M. 2001. Adjustable autonomy in real-world multi-agent environments. In *Proceedings of the Conference on Autonomous Agents*, to appear.

Sun Microsystems, Inc. 1999. Jini architectural overview. `http://www.sun.com/jini/whitepapers/architecture.pdf`.

Sycara, K., and Zeng, D. 1994. Towards an intelligent electronic secretary. In *CIKM-94 (International Conference on Information and Knowledge Management), Intelligent Information Agents Workshop*.

Sycara, K., and Zeng, D. 1996. Coordination of multiple intelligent software agents. *International Journal of Cooperative Information Systems* 5(2&3).

Sycara, K.; Decker, K.; Pannu, A.; Williamson, M.; and Zeng, D. 1996. Distributed intelligent agents. *IEEE Expert*.

Sycara, K.; Lu, J.; Klush, M.; and Widoff, S. 1999. Matchmaking among heterogeneous agents in the internet. In *AAAI Spring Symposium on Intelligent Agents in Cyberspace*.

Tambe, M. 1997. Towards flexible teamwork. *Journal of Artificial Intelligence Research* 7:83–124.

w. R. Swartout, and Gil, Y. 1995. Expect: Explicit representations for flexible acquisition. In *Proc. Ninth Knowledge Acquisition for Knowledge-Based Systems Workshop*.