From: IAAI-01 Proceedings. Copyright © 2001, AAAI (www.aaai.org). All rights reserved.

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

Many activities of a human organization are 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 could assist all organizations, including disaster response organizations, corporations, the military, universities and research institutions.

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.

Copyright © 2001, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

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 more 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 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

Our agents help coordinate the everyday activities of a research project: they 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: A Palm VII PDA with GPS receiver

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 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 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. We have begun relying on our agents so heavily to order lunch that one local "Subway" restaurant owner even remarked: *"...more and more computers are getting to order food...so we might have to think about marketing [to them]."*

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 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.

To identify individuals with related interests, the agents use an online bibliography service that provides a list 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 going to the same meeting and then build a model of the research interests of the different participants based on their publications. Using this information, the user selects the participants for the meeting and the agent sends out an invitation to each of the potential attendees.

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

Electric Elves agents 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). On the other hand, it may act without autonomy, instead 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). One problem became apparent when applying this technique in Electric Elves: a user would not grant autonomy to Friday in making certain decisions, but s/he would sometimes be unavailable to provide any input at decision time. Thus, a Friday could end up waiting indefinitely for user input and miscoordinate with its teammates.

We therefore modified the system so that if a user did not respond within a fixed time limit, Friday acted autonomously based on 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 erroneously volunteered him to give a presentation. C4.5 had overgeneralized 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, but it instead flexibly reevaluates when its user does not respond, sometimes reversing its decision and taking back autonomy; (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 actions and plan a policy that maximizes team welfare.

We have 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". If the user does not arrive by the time of the meeting, the policy 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.

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 of what other agents are available and how they are invoked.

We have developed an agent matchmaker called PHOSPHORUS (Gil & Ramachandran 2001), which builds on previous research on matching problem solving goals and methods in EXPECT (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.

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 translated into Loom (MacGregor 1991). Loom's classifier recognizes that the capability to "setup equipment" will subsume one to "setup LCD projector", because according to the domain ontologies equipment subsumes LCD projector.

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 if one required a single agent to match all capabilities in the request. PHOSPHORUS supports set reformulations (breaking down a task on a set into its individual elements) and covering reformulations (decomposing a task into the disjoint subclasses of its arguments). For example, no single agent can discuss the entire Electric Elves project, since no single researcher is involved in all the aspects of the project. But PHOSPHORUS can return a set of people who can collectively cover the topic 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)
```

Many additional challenges lay ahead regarding capability representations for people within the organization. For example, although anyone has the capability to call a taxi for a visitor (and will do so if necessary), project assistants are the preferred option. Extensions to the language are 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, including organizational databases (personal schedules, staff lists) and external Web sites, such as airline schedules, restaurant information, traffic and weather updates, etc. In order to pick a restaurant for a scheduled lunch meeting, the agents access the Restaurant Row site to get the locations of restaurants that meet the specified criteria, e.g., dietary restrictions. Wrappers enable Web sources to be queried as if they were databases by other applications, such as the Electric Elves agents. A critical part of a wrapper is a set of extraction rules that enable the wrapper to quickly locate the beginning and end of the data to be extracted from a Web page in response to some query.

The Ariadne component (Knoblock *et al.* 2000; 2001) of Electric Elves learns wrappers from pages in which relevant data has been labeled by the user. Previous research has focused on applying machine learning techniques to rapidly generate wrappers (Muslea, Minton, & Knoblock 2000; Kushmerick 2000), but few attempts have been made to validate data, detect failures (Kushmerick 1999) or repair wrappers when the source pages change in a way that breaks the wrapper. Automatically monitoring external information sources and repairing wrappers when errors are detected is a critical part 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 content of the extracted data. Since the information for a single data 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, we can launch the wrapper repair process.

The learned patterns represent the structure of data as a sequence of words and wildcards. Wildcards represent syntactic categories to which words belong—alphabetic, numeric, capitalized, *etc.*. For example, a set of street addresses all start with a pattern "*_Number_ _Capitalized_*": a number 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 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), but our pattern language is better suited for capturing the regularities in small data fields (as opposed to languages). For verification, we learn the patterns from training examples (data extracted by the wrapper that is known to be correct). Next, the wrapper generates a set of test examples from pages retrieved using the same or similar set of queries. If the patterns describe statistically the same proportion of the test examples as the training examples, the wrapper is deemed correct; otherwise, it has failed.

The most common causes of wrapper failure are changes in Web site layout. Even minor changes can break the wrapper's data extraction rules. However, since the content tends to remain the same, it is often possible to automatically repair the wrapper by learning new extraction rules. We exploit the learned patterns to find correct examples of data on the new pages. The Restaurant Row wrapper allows us to retrieve several examples of restaurant addresses, and the verification algorithm learned that some of the examples start with the pattern "*_Number_ _Capitalized_*" and end with the pattern "Avenue". If Restaurant Row changes to look more like the Zagat Web site, the wrapper will no longer extract addresses correctly. In the verification phase we will detect the failure because the extracted data is not described by the patterns. However, since restaurant addresses still start with "*_Number_ _Capitalized_*" and end with "Avenue", we should be able to find addresses on the changed pages. Once the desired information has been found, these examples and the new pages are sent to the wrapper generation system to learn new data extraction rules. We use prior knowledge about the content of data, as captured by the learned patterns, along with *a priori* expectations about the data to identify correct examples on the changed pages. We can 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.

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 yellow pages-type source, and use it to create a wrapper for this source.

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 in which they operate. Some of this information can be provided directly from existing databases and online sources, but other information—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 relatively static information. 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 combined statistical match techniques from the area of information retrieval (IR) with logic-based matching performed by the PowerLoom knowledge representation (KR) system. The IR techniques work well with unstructured text sources available online on the Web, which is the form in which information is typically available to outside organizations. PowerLoom facilitates declarative modeling of the decision process, modeling of missing information, logical inference, explanation and also customization.

The matchmaker's knowledge base contains an ontology of research topic areas and associated relations; rules formalizing the matchmaking process; and 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, the following query finds candidates for lunch with Yolanda Gil:

```
(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)))))
```

For more specific purposes, any of the more basic relations comprising should-meet such as interests-overlap could be queried directly by a client. Using a general purpose KR system as the matching engine provides us with this flexibility. 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 deal with incompleteness of the KB, we allow a requesting agent to introduce new individuals and then the Interest Matcher automatically infers limited structured knowledge—their research interests—by analyzing relevant unstructured text sources on the Web.

The key idea is that people's research interests are implicitly documented in their publication record. We make these interests explicit by associating each research topic in the PowerLoom topic ontology with a statistical representation 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. 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).

Coordination of Component Agents

The diverse agents in Electric Elves must work together to accomplish the complex tasks of the whole system. For instance, to plan a lunch meeting, the interest matcher finds a list of potential attendees, the Friday of each potential attendee decides whether s/he will attend, the capability matcher identifies dietary restrictions of the confirmed attendees, and the reservation site wrapper identifies possible restaurants and makes the final reservation. In addition to low-level communication issues, there is the complicated problem of getting all these agents to work together as a team. Each of these agents must execute its part in coordination with the others, so that it performs its tasks at the correct time and sends the results to the agents who need them.

However, constructing teams of such agents remains a difficult challenge. 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. The absence of such teamwork models makes team construction highly labor-intensive. Human developers must provide the agents with a large number of problem-specific coordination and communication plans that are not reusable. 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., a presenter is delayed), members' divergent beliefs, and unexpectedly noisy communication. It is difficult to anticipate and preplan for all possible coordination failures.

In Electric Elves, the agents coordinate using Teamcore, a domain-independent, decentralized, teamwork-based integration architecture (Pynadath *et al.* 1999). Teamcore uses STEAM, a general-purpose teamwork model (Tambe 1997) and provides core teamwork capabilities to agents by wrapping them with Teamcore proxies (separate from the Friday agents that are *user* proxies). By interfacing with Teamcore proxies, existing agents can rapidly assemble themselves into a team to solve a given problem. The Teamcore proxies form a distributed *team-readiness* layer that provides 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 all five required social capabilities.

Each and every agent in the Electric Elves organization (Fridays, matchers, wrappers) has an associated Teamcore proxy that records its membership in various teams and active commitments made to these teams. Given an abstract specification of the organization and its plans, 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 and robust plan execution. 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).

Electric Elves Architecture

Electric Elves is a complex and heterogeneous system spanning a wide variety of component technologies and languages, communication protocols as well as operating system platforms. Figure 2 shows the components of the current version of 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. Initially we looked for an implementation of KQML, but there was none available that supported all the languages and platforms we required. To solve this integration problem, we are using the DARPA supported CoABS Grid technology developed by Global InfoTek, Inc. and ISX Corporation[1]. The CoABS Grid is a Java-based communication infrastructure built on top of Sun's Jini networking technology. 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. Grid proxy components connect non-Java technology to the Grid.

We primarily use the CoABS Grid as a uniform transport mechanism. The content of Grid messages 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.

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 (Lesser *et al.* 1999). These projects are embedding sensor networks and agents to control and improve our everyday physical environments. This kind of infrastructure would make it easier for Electric Elves to locate and contact people as well as to direct the environmental control agents in support of 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 (with interruptions 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: workstation display, voice, mobile phones, and palm pilots. They

[1] http://coabs.globalinfotek.com/coabs_public/coabs_pdf/gridvision.pdf

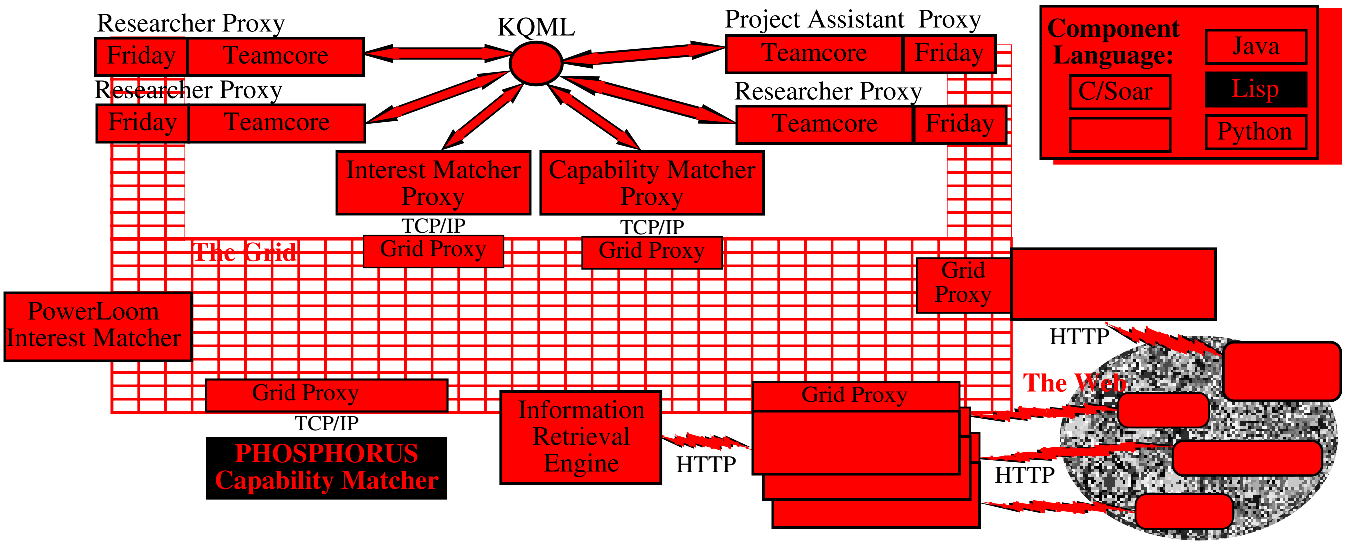

Figure 2: Electric Elves System Architecture

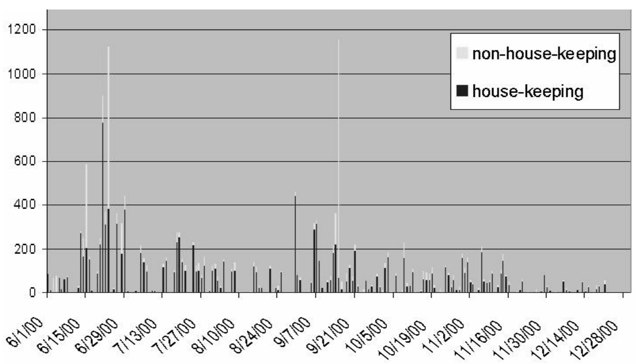

Figure 3: Number of daily coordination messages exchanged by proxies over a seven-month period.

also communicate with restaurants by sending faxes.

Figure 3 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.

Several observations show the effectiveness of Electric Elves. 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. Also the overhead of waiting for delayed members in meeting rooms has been reduced. Overall, 1128 meetings have been monitored, out of which 285 have been rescheduled, 230 automatically and 55 by hand. Both autonomous rescheduling and human intervention were useful in Elves.

Furthermore, whereas in the past, one of our 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 receives notification of any meeting requiring a presentation. Auction decisions may be made without requiring a full set of bids; in fact, in one case, only 4 out of 9 possible bids were received. The rest of the group simply did not bid before the winner was announced. Most of the time, the 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 easily accommodates.

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.

Acknowledgements

The research reported here was supported in part by the Rome Laboratory of the Air Force Systems Command and the Defense Advanced Research Projects Agency (DARPA) under contract numbers F30602-97-C-0068, F30602-98-2-0109 and F30602-98-2-0108; and in part by the Air Force Office of Scientific Research under Grant Number F49620-01-1-0053. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

Surya Ramachandran also contributed to the success of this project.

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 *Proc. 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 *Proc. of AAAI-1992*, 96–103.

Dorais, G. A.; Bonasso, R. P.; Kortenkamp, D.; Pell, B.; and Schreckenghost, D. 1998. Adjustable autonomy for human-centered autonomous systems on Mars. In *Proc. 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 *Proc. of the 3rd Conf. on Artificial Intelligence Planning Systems*, 70–77.

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. PHOSPHORUS: A task-based agent matchmaker. In *Proc. of the Fifth International Conference on Autonomous Agents*.

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

Kautz, H.; Selman, B.; Coen, M.; and Ketchpel, S. 1994. An experiment in the design of software agents. In *Proc. of AAAI-1994*.

Knoblock, C. A.; Lerman, K.; Minton, S.; and Muslea, I. 2000. Accurately and reliably extracting data from the web: A machine learning approach. *Data Engineering Bulletin* 23(4).

Knoblock, C. A.; Minton, S.; Ambite, J. L.; Ashish, N.; Muslea, I.; Philpot, A. G.; and Tejada, S. 2001. The ARIADNE approach to web-based information integration. *International Journal of Cooperative Information Systems* 10(1/2).

Kushmerick, N. 1999. Regression testing for wrapper maintenance. In *Proc. of AAAI-1999*, 74–79.

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

Lerman, K., and Minton, S. 2000. Learning the common structure of data. In *Proc. of 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 *Proc. of the 3rd Annual Conf. on Autonomous Agents*, 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. 2000. Hierarchical wrapper induction for semistructured information sources. *Journal of Autonomous Agents and Multi-Agent Systems* 4(1/2).

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 *Proc. of the Conference on Autonomous Agents*.

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

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

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