Adjustable Autonomy for the Real World

Milind Tambe, Paul Scerri, David V. Pynadath
Information Sciences Institute and Computer Science Department
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{tambe,scerri,pynadath}@isi.edu

Abstract

Adjustable autonomy refers to agents' dynamically varying their own autonomy, transferring decision making control to other entities (typically human users) in key situations. Determining whether and when such transfers of control must occur is arguably the fundamental research question in adjustable autonomy. Previous work, often focused on individual agent-human interactions, has provided several different techniques to address this question. Unfortunately, domains requiring collaboration between teams of agents and humans reveals two key shortcomings of these previous techniques. First, these techniques use rigid one-shot transfers of control that can result in unacceptable coordination failures in multiagent settings. Second, they ignore costs (e.g., in terms of time delays or effects of actions) to an agent's team due to such transfers of control.

To remedy these problems, this paper presents a novel approach to adjustable autonomy, based on the notion of *transfer of control strategy*. A transfer of control strategy consists of a sequence of two types of actions: (i) actions to transfer decision-making control (e.g., from the agent to the user or vice versa) (ii) actions to change an agent's pre-specified coordination constraints with others, aimed at minimizing miscoordination costs. The goal is for high quality individual decisions to be made with minimal disruption to the coordination of the team. These strategies are operationalized using Markov Decision Processes to select the optimal strategy given an uncertain environment and costs to individuals and teams. We present a detailed evaluation of the approach in the context of a real-world, deployed multi-agent system that assists a research group in daily activities.

Introduction

Exciting, emerging applications ranging from intelligent homes (Lesser *et al.* 1999), to "routine" organizational coordination (Pynadath *et al.* 2000), to electronic commerce (Collins *et al.* 2000), to long-term space missions (Dorais *et al.* 1998) utilize the decision making skills of both agents and humans. Such applications have fostered an interest in *adjustable autonomy* (AA), which allows an agent to dynamically change its own autonomy, transferring control for some of its key decisions to humans or other agents(for Papers 1999). With AA, an agent need not make all the decisions autonomously; rather, it can choose to reduce its own autonomy and let users or other agents make some decisions.

A central problem in AA is to determine whether and when transfers of decision-making control should occur. The key challenge here is to balance two potentially conflicting goals. First, to ensure that the highest quality decisions are made, the agent must transfer control to the human user (or other agents) whenever they provide superior decision making expertise. On the other hand, interrupting a human user has very high costs and may fail for a variety of reasons, and thus such transfers of control must be minimized. Previous work provides several different techniques that attempt to balance these two conflicting goals and thus address the transfer of control problem. For example, one technique suggests that decision-making control should be transferred if the expected utility of doing so is higher than the expected utility of keeping control over the decision(Horvitz, Jacobs, & Hovel 1999). A second technique uses uncertainty as the rationale for deciding who should have control, forcing the agent to relinquish control to the human whenever uncertainty is high(Gunderson & Martin 1999). Other techniques transfer control if any incorrectness in an agent's autonomous decision can cause significant harm(Dorais *et al.* 1998) or if the agent lacks the capability to make the decision(Ferguson, Allen, & Miller 1996).

Unfortunately, these transfer-of-control techniques and indeed most previous work in AA, have been focused on single-agent and single-human interactions. When applied to interacting teams of agents and humans, or multiagent settings in general, these techniques lead to dramatic failures. In particular, they fail to address a key requirement in multiagent settings, that of ensuring joint or coordinated actions (in addition to balancing the two goals already mentioned above). They fail because they ignore team related factors, such as costs the to the team due to delays in decisions, during such transfers of control. More importantly, these techniques use one-shot transfers of control, rigidly committing to one of two choices: (i) transfer control to a human and wait for human input (choice H) or (ii) do not transfer control and take autonomous action (choice A). However, given interacting teams of agents and humans, either choice can lead to significant coordination failures if the entity in control cannot provide the relevant decision in time for the co-

ordinated action. On the other hand, if the agent commits to one of choices simply to avoid miscoordination, that can result in costly errors. As an example, consider an agent that manages an individual user's calendar and can request the rescheduling of a team meeting if it thinks the user will be unable to attend on time. Rescheduling is costly, because it disrupts the calendars of the other team members, so the agent can ask its user for confirmation to avoid making an unnecessary rescheduling request. However, while it waits for a response, there is miscoordination with other users. These other users will begin arriving at the meeting room and if the user does not arrive, then the others will waste their time waiting as the agent sits idly by. On the other hand, if, despite the uncertainty, the agent acts autonomously and informs the others that the user cannot attend, then its decision may still turn out to be a costly mistake. Indeed, as seen in Section 2, when we applied a rigid transfer of control decision-making to a domain involving teams of agents and users, it failed dramatically.

Yet, many emerging applications do involve multiple agents and multiple humans acting cooperatively towards joint goals. To address the shortcomings of previous AA work in such domains, this article introduces the notion of *transfer of control strategies*. A transfer of control strategy consists of a planned sequence of two types of actions: (i) actions to transfer decision-making control (e.g., from the agent to the user or vice versa) (ii) actions to change an agent's pre-specified coordination constraints with others, postponing or reordering activities as needed (typically to buy time for the required decision). The agent executes such a strategy by performing the actions in sequence, transferring control to the specified entity and changing coordination as required, until some point in time when the entity currently in control exercises that control and makes the decision. Thus, the previous choice of H or A are just two of many different and possibly more complex transfer-of-control strategies. For instance, an ADH strategy implies that an agent A, initially attempts autonomous actions given a problem. If the agent A makes the decision, the strategy execution ends there. However, there is a chance that it is unable to take that action in a timely manner, perhaps because a web server it relies on is down. In this case, it executes D, to delay the coordinated action it has planned with others, and thus eliminate or reduce any miscoordination costs. D has the effect of "buying time" to provide the human H more time to make the decision, and thus reduce decision uncertainty. The agent then transfers control to a human user (H). In general, if there are multiple decision-making entities, say one agent and two separate human users H_1 and H_2, a strategy may involve all of them, e.g., H_1AH_2. While such strategies may be useful in single-agent single-human interactions, they are particularly critical in general multiagent settings, as discussed below.

Such strategies provide a flexible approach to transfer-of-control in complex systems with many actors. By enabling multiple transfers of control between two (or more) entities, rather than rigidly committing to one entity (i.e., A or H), a strategy attempts to provide the highest quality decision, while avoiding coordination failures. In particular, there is uncertainty about which entity will make that decision and when it will do so, e.g., a user may fail to respond, an agent may not be able to make a decision as expected or other circumstances may change. A strategy addresses such uncertainty by planning multiple transfers of control to cover for such contingencies. For instance, with the ADH strategy, an agent ultimately transfers control to a human to ensure that some response will be provided in case the agent fails to act. Furthermore explicit coordination change actions, such as D, reduce miscoordination costs while better decisions are being made. These strategies must be planned: often, a sequence of coordination changes may be needed, and since each coordination change is costly, agents need to look ahead at possible sequences of coordination changes, selecting one that maximizes team benefits.

The key question in transfer of control is then to select the right strategy, i.e., one that optimizes all of the different costs and benefits: provide the benefit of high-quality decisions without risking significant costs in interrupting the user and miscoordinating with the team. Furthermore, an agent must select the right strategy despite significant uncertainty. Markov decision processes (MDPs)(Puterman 1994) are a natural choice for implementing such reasoning because they explictly represent costs, benefits and uncertainty as well as doing lookahead to examine sequences of actions.

Our research has been conducted in the context of a real-world multi-agent system, called *Electric Elves* (E-Elves) (Pynadath *et al.* 2000), that we have used for several months at USC/ISI. E-Elves assists a group of researchers and a project assistant in their daily activities, providing a unique, exciting opportunity to test ideas in a real environment. Individual user proxy agents called Friday (from Robinson Crusoe's servant Friday) assist with rescheduling meetings, ordering meals, finding presenters and other day to day activities. Over the course of several months MDP based AA reasoning was used around the clock in the E-Elves making many thousands of autonomy decisions. Despite the unpredictability of the users and limited sensing abilities, the autonomy reasoning consistently produced reasonable results. Many times the agent performed several transfers of control to cope with contingencies such as a user not responding. Detailed experiments verify that the MDP balanced the costs of asking for input, the potential for costly delays and the uncertainty in a domain when doing the autonomy reasoning.

Adjustable Autonomy – The Problem

We consider the general problem of AA in a team context as follows. The team, which may consist entirely of agents or include humans, has some joint activity, α. The agent has a role, ρ, in the team. Coordination constraints exist between ρ and the roles of other members of the team. For example, various roles might need to be executed simultaneously or in a certain order or with some combined quality. Maintainance of the constraints is necessary for the success of the joint activity. The primary goal of the agent is to ensure the successful completion of the joint activity, α, via fulfillment of the role, ρ. Performing the role requires that one or more non-trivial decisions be made. The agent can

transfer decision-making control for a decision to another agent or user (outside of the team), thereby reducing its autonomy. Different agents and users will have differing abilities to make the decisions due, for example, to available computational resources or access to relevant information. The agent may fulfill ρ either through decisions it makes itself or by transferring control to another human or agent to make the decision. It should do so whenever it reasons that doing so will be in the best interests of the joint activity.

Given the multi-agent context, a critical facet of the successful completion of the joint task is to ensure that coordination between team members is maintained. Miscoordination between team members may occur for a variety of reasons, though here we are primarily concerned with mis-coordination due to delays in a decision being made. From the perspective of the AA, the agent must ensure that transfers of control do not lead to delays that in turn lead to miscoordination. For example, delays might occur because the user or agent to which control is transferred is otherwise occupied or can not be contacted or it may occur because making the decision takes longer than expected. When the agent transfers control it does not have any guarantee on the timeliness or quality of the decision made by the entity to which control is transferred. In fact, in some cases it will not know whether the entity will be able to make a decision at all or even whether the entity will know it has decision making control.

To avoid mis-coordination an agent can request that coordination constraints be changed to allow more time for a decision to be made. A coordination change might simply involve reordering or delaying tasks or it may be a more dramatic change where the team uses a completely different approach to reach its goal. While changing coordination constraints is not a desirable action per se, it is better than miscoordination. Hence, changes in coordination should only be made if the potential value of the extra time made available for the decision outweighs the cost of that change. It is possible that when an agent requests a coordination change, the team can decide to deny the request. For example, the agent may request a change that from its local perspective is of low cost but another team member might have information that the change will actually cause a complete failure, hence the request for a coordination change might be rejected. Despite the ability for the team to deny the request the agent should act responsibly and not burden the team with unnecessary requests.

The Electric Elves

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. USC/ISI is taking the first step to realizing this vision with the Electric Elves (E-Elves). The E-Elves provide a unique opportunity to do research on AA. General ideas and techniques can be

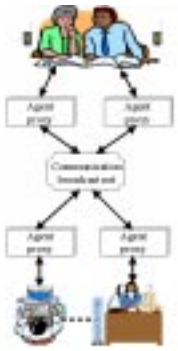


Figure 1: Overall proxy-based architecture

implemented and tested in a real world system, allowing the strengths and weaknesses of those approaches to be examined objectively. Moreover, having a concrete application allows for a more accessible discussion of abstract problems and solutions.

Tied to individual user workstations, fax machines, voice and mobile devices such as cell phones and palm pilots, E-Elves assist in routine tasks, such as rescheduling meetings, selecting presenters for research meetings, monitoring flight departure times, tracking people's locations, organizing lunch meetings, etc.(Chalupsky *et al.* 2001) A number of underlying AI technologies that support the E-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(Pynadath *et al.* 2000). While all these technologies are interesting, here we focus on the AA technology.

(a) **(b)**

Figure 2: (a) Friday asking the user for input regarding ordering a meal. (b) Palm VII and GPS

The overall design of the E-Elves is shown in Figure . Each proxy is called Friday and acts on behalf of its user in the agent team. The basic design of the Friday proxies is discussed in detail elsewhere (Tambe *et al.* 2000) (where they are referred to as TEAMCORE proxies). Currently, Friday can perform a variety of tasks for its user. If a user is delayed to a meeting, Friday can reschedule the meeting, informing other Fridays, who in turn inform their human users. If there is a research presentation slot open, Friday may respond to the invitation to present on behalf of its user. Friday can also

order its user's meals (see Figure 2(a)) and track the user's location, posting it on a Web page. Friday communicates with users using wireless devices, such as personal digital assistants (PALM VIIs) and WAP-enabled mobile phones, and via user workstations. Figure 2(b) shows a PALM VII connected to a Global Positioning Service (GPS) device, for tracking users' locations and enabling wireless communication with Friday.

Each Friday's team behavior is based on a teamwork model, called STEAM(Tambe 1997). Friday models each meeting as a team's joint intention that, by the rules of STEAM, they keep each other informed about (e.g., a meeting is delayed, cancelled, etc.). Furthermore, Fridays use STEAM role relationships to model the relationships among team members. For instance, the presenter role is critical since the other attendees depend on someone giving a presentation. Thus, if the presenter cannot attend, the team recognizes a critical role failure that requires remedial attention.

AA is used for several decisions in the E-Elves, including closing auctions for team roles, ordering lunch and rescheduling meetings. AA is important since clearly the user has all the critical information pertaining to the decisions and hence good decisions will sometimes require user input. The decision on which we focus is ensuring the simulataneous arrival of attendees at a meeting. If any attendee arrives late, or not at all, to the meeting, the time of all the attendees is wasted, yet delaying the meeting is disruptive to the user's schedules. Friday acts as proxy for its user, hence its responsibility is to ensure that its user arrives at the meeting at the same time as other users. The decision that the agent has responsibility for making is whether the user will arrive at the meeting's currently scheduled time. Clearly, the user will be often better placed to make this decision. However, if the agent transfers control to the user for the decision it must guard against mis-coordination while the user is responding. Although the user will not take long to make the decision, it may take a long time to contact the user, e.g., if the user is in another meeting. If the user is contacted they can decide to delay the meeting to a time when they will be able to arrive simulataneously with other users or inform the team they will not arrive at all (allowing other attendees to proceed without them). Although the agent can make the same decisions as the user, to decide that the user is not attending is costly and the agent should avoid deciding on it autonomously. Thus, unless it is certain that a potentially costly action is the correct one to take, the agent should try to transfer control to the user rather than acting autonomously. To buy more time for the user to make the decision the agent could delay the meeting, i.e., do a coordination change. So the agent has several options, i.e., autonomous decision, transfer of control and coordination change, and a variety of competing influences, e.g., not wasting the user's time and not making mistakes, that need to be balanced by its autonomy reasoning.

Decision-tree approach

Whilst the problem of AA in a team context is intuitively more subtle than that of the single agent - single human case, care needs to be taken not to develop new complex, solutions when simpler, previously reported solutions would suffice. To this end, a first attempt was inspired by CAP (Mitchell *et al.* 1994), an agent system for helping a user schedule meetings. Like CAP, Friday learned user preferences using C4.5 decision-tree learning (Quinlan 1993). The key idea was to resolve the transfer-of-control decision by learning from user input. In training mode, Friday recorded values of a dozen carefully selected attributes and the user's preferred action (identified by query via a dialog box, as in Figure) whenever it had to make a decision. Friday used the data to learn a decision tree (e.g., *if* the user has a meeting with his or her advisor, but is not at ISI at the meeting time, *then* delay the meeting 15 minutes). Also in training mode, Friday asked if the user wanted such decisions taken autonomously in the future. Friday again used C4.5 to learn a second decision tree from these responses.

Initial tests with the above setup were promising (Tambe *et al.* 2000), but a key problem soon became apparent. When Friday encountered a decision it learned not take autonomously, it would wait indefinitely for the user to make the decision, even though this inaction led to miscoordination with teammates. To address this problem, if a user did not respond within a fixed time limit, Friday took an autonomous action. Although results improved, when the resulting system was deployed 24/7, it led to some dramatic failures, including:

1. Tambe's (a user) Friday incorrectly, autonomously cancelled a meeting with the division director. C4.5 over-generalized from training examples.
2. Pynadath's (another user) Friday incorrectly cancelled the group's weekly research meeting. A time-out forced the choice of an (incorrect) autonomous action when Pynadath did not respond.
3. A Friday delayed a meeting almost 50 times, each time by 5 minutes. The agent was correctly applying a learned rule but ignoring the nuisance to the rest of the meeting participants.
4. Tambe's proxy automatically volunteered him for a presentation, though he was actually unwilling. Again, C4.5 had overgeneralized from a few examples and when a timeout occurred had taken an undesirable autonomous action.

Some failures were due to the agent making risky decisions despite considerable uncertainty because the user did not quickly respond (examples 2 and 4). Other failures were due to insufficient consideration being given to team costs and the potential for high team costs due to incorrect actions. Yet, other failures could be attributed to the agent not planning ahead adequately. For instance, in example 3, each five-minute delay is appropriate *in isolation*, but the rules did not consider the ramifications of one action on successive actions. Planning could have resulted in a one-hour delay instead of many five-minute delays. From the growing list of failures, it became clear that the approach faced some significant problems. While the agent might have eventually been able to learn rules that would successfully balance all the costs and deal with all the uncertainty and handle all the special cases and so on, a very large amount of training data would be required, even for this relatively simple decision. Hence, while the C4.5 approach worked in the single

agent - single human context, AA in the team context has too many subtleties and too many special cases for the agent to learn appropriate actions with a reasonable amount of training data. Furthermore, the fixed timeout strategy constrained the agent to certain sequences of actions, limiting its ability to deal flexibly with changing situations.

Flexible Transfer of Control via MDPs

In a multi-agent domain to avoid miscoordination we must design agents centered around the notion of a *transfer-of-control strategy*. A transfer-of-control strategy is a planned sequence of transfer-of-control actions, which include both those that actually transfer control and those that change coordination constraints to buy more time to get input. The agent executes such a strategy by performing the actions in sequence, transferring control to the specified entity and changing coordination as required, until some point in time when the entity currently in control exercises that control and makes the decision. More precisely, we consider a scenario where an agent, A, is responsible for making a decision, d. The agent can draw upon n other entities from a set $E = \{e_1 \dots e_n\}$, who are all capable (perhaps unequally) of making decision d instead. The entities can be either humans or other agents. Agent A can transfer decision-making control to any entity e_i, and we denote such a transfer-of-control action with the symbol e_i. In the typical AA setting, the agent A is itself one of the available decision-making entities.

For the purposes of this discussion, we assume that the agent can make the decision instantaneously (or at least, with no delay significant enough to affect the overall value of the decision). On the other hand, the other entities may *not* make the decision instaneously, e.g., a human user may not be able to respond immediately. Therefore, when the agent transfers decision-making control to another entity, it may stipulate a limit on the time that it will wait for a response from that entity. To capture this additional stipulation, we denote transfer-of-control actions with this time limit as an action $e_i(t)$, i.e., e_i has decision-making control for a maximum time of t. Such an action has two possible outcomes: either e_i responds before time t and makes the decision, or else it does not respond and decision d remains unmade at time t. As an illustration, consider the E-Elves, where there are two entities: the human user, H, and the agent, A. The action, $H(5)$, would denote asking the user for input and waiting at most 5 minutes before timing out on the query.

In addition, the agent has some mechanism by which it can take a *deadline-delaying action* (denoted $\mathcal{D}$) to alleviate any temporal pressures on the decision. A $\mathcal{D}$ is a generalization of the "delay meeting" action from the E-Elves. The $\mathcal{D}$ action has an associated value, $\mathcal{D}_{value}$, which specifies its magnitude, i.e., how much the $\mathcal{D}$ has alleviated the temporal pressure.

We can concatenate these individual transfer-of-control actions to produce a strategy. The agent then executes the sequence of transfer-of-control actions in the sequence, halting whenever the entity in control responds. For instance, in the E-Elves, the strategy $H(5)A$ would specify that the agent first give up control and ask the human user. If the human responds with a decision within 5 minutes, then the task is complete. If not, then the agent proceeds to the next transfer-of-control action in the sequence. In this example, this next action, A, specifies that the agent itself make the decision and complete the task. We can define the space of all possible strategies as follows:

$$\mathbf{S} = (E \times \mathcal{R}) \times ((E \times \mathcal{R}) \cup \{\mathcal{D}\}) \quad (1)$$

For readability, we will frequently omit the time specifications from the transfer-of-control actions and instead write just the order in which the agent transfers control among the entities and executes $\mathcal{D}$s (e.g., HA instead of $H(5)A$). Thus, this shorthand does not record the timing of the transfers of control. Using this shorthand, we will focus on a smaller space of possible transfer-of-control strategies:

$$\mathbf{S} = E \times (E \cup \{\mathcal{D}\}) \quad (2)$$

This space of strategies provides an enormous set of possible behavior specifications. The agent must select the strategy that maximizes the overall utility of the eventual decision. Presumably, each entity has different decision-making capabilities; otherwise, the choice among them has little impact. We model each entity as making the decision with some expected quality, $\mathbf{EQ} = \{EQ_e^d : \mathcal{R} \rightarrow \mathcal{R}\}_{i=1}^n$. The agent knows $\mathbf{EQ}$, perhaps with some uncertainty.

In addition, when given decision-making control, each entity may have different response times. The functions, $\mathbf{P} = \{P_\top(t) : \mathcal{R} \rightarrow [0,1]\}$, represent continuous probability distributions over the time that the entity in control will respond with a decision of quality $EQ_e^d(t)$. In other words, the probability that e_i will respond within time t_0 is $P_\top(t_0)$.

The agent and the entities are making decisions within a dynamic environment, and in most real-world environments, there are time pressures. We model this temporal element through a *wait-cost function*, $\{\mathcal{W} : t \rightarrow R\}$, that represents the cost of delaying a decision until time t. The set of possible wait-cost functions is $\mathbf{W}$. We assume that $\mathcal{W}(t)$ is non-decreasing and that there is some point in time, $\lhd$, beyond which there is no further cost of waiting (i.e., $\forall t \geq \lhd, \forall \mathcal{W} \in \mathbf{W}, \mathcal{W}(t) = \mathcal{W}(\lhd)$). The deadline-delaying action moves the agent further away from this deadline and reduces the rate at which wait costs are accumulating. We model the value of the $\mathcal{D}$ by letting $\mathcal{W}$ be a function of $t - \mathcal{D}_{value}$ (rather than t) after the $\mathcal{D}$ action. Presumably, such delays do not come for free, or else the agents could postpone the decision indefinitely to no one's loss. We model the $\mathcal{D}$ as having a fixed cost, $\mathcal{D}_{cost}$, incurred immediately upon its execution.

We can use all of these parameters to compute the expected utility of a strategy, s. The uncertainty arises from the possibility that an entity in control may not respond. The strategy specifies a contingent plan of transfer-of-control actions, where the agent executes a particular action contingent on the lack of response from all of the entities previously given control. The agent derives utility from the decision eventually made by a responding entity, as well as from the costs incurred from waiting and from delaying the deadline. The problem for the agent can then be defined as:

Definition 3.1 *For a decision d, the agent must select $s \in \mathbf{S}$ such that $\forall s' \in \mathbf{S}, s' \neq s, EU_s^d(t) \geq EU_{s'}^d(t)$*

MDP-based Evaluation of Strategies

MDPs are a natural mechanism for choosing a transfer of control strategy that maximizes expected utility. By encoding the transfer-of-control actions and the associated costs and utilities within an MDP, we can use standard algorithms (Puterman 1994) to compute an optimal policy of action that maps the agent's current state into the optimal action for that state. We can then interpret this policy as a transfer-of-control strategy.

In representing the state of execution of a transfer-of-control strategy, the key feature is the e_i*-response*, a variable indicating the response (if any) of e_i. The state must also represent various aspects of the decision d, which, in the E-Elves, concerns a team activity, α, and the user's role, ρ, within α. Thus, the overall state, within the MDP representation of a decision d, is a tuple:

$<$ *team-orig-expect-*ρ, *team-expect-*ρ, *agent-expect-*ρ, α*-status*, e_i*-response*, other α attributes $>$

Here, *team-expect-*ρ is the team's current expectations of what fulfilling the role ρ implies, while *team-orig-expect-*ρ is what the team originally expected of the fulfilling of the role. Similarly, *agent-expect-*ρ is the agent's (probabilistic) estimation for how ρ will be fulfilled. For example, for a meeting scenario, *team-orig-expect-*ρ could be "Meet at 3pm", *team-expect-*ρ could be "Meet at 3:15pm" after a user requested a delay and *agent-expect-*ρ could be "Meet at 3:30pm" if it believes its user will not make the resecheduled meeting.

We can specify the set of actions for this MDP representation as $\Gamma = E \cup \{\mathcal{D}, \text{wait}\}$. The set of actions subsumes the set of entities, E, since the agent can transfer decision-making control to any one of these entities. The $\mathcal{D}$ action is the deadline-delaying action as discussed earlier. The "wait" action puts off transferring control and making any autonomous decision, without changing coordination with the team. The agent should reason that "wait" is the best action when, in time, the situation is likely to change to put the agent in a position for an improved autonomous decision or transfer of control, without significant harm to the team-level coordination relationships.

The transition probabilities represent the effects of these actions as a distribution over their effects (i.e., ensuing state of the world). When the agent chooses an action that transfers decision-making control to an entity other than the agent itself, there are two possible outcomes: either the entity makes a decision (producing a terminal state), or the decision remains unmade (the result being as if the agent had simply waited). We compute the relative likelihood of these two possible transitions by using the response times modeled in $\mathbf{P}$. The $\mathcal{D}$ action has a deterministic effect, in that it changes the coordination of α (affecting the expectations on the user's role through the state feature, *team-expect-*ρ).

The final part of our MDP representation is the reward function. In general, our AA MDP framework uses a reward function:

$$R(s, a) \tag{3}$$

Figure 3: Dialog box for delaying meetings.

$$\begin{aligned} &= f(\textit{team-orig-expect-}\rho(s), \textit{team-expect-}\rho(s), \\ &\qquad \textit{user-expect-}\rho(s), \alpha\textit{-status}(s), a) \\ &= -\lambda_1 f_1(\| \textit{ team-orig-expect-}\rho(s) - \textit{team-expect-}\rho(s) \ \|) \\ &\quad -\lambda_2 f_2(\| \textit{ team-expect-}\rho(s) - \textit{user-expect-}\rho(s) \ \|) \\ &\quad +\lambda_3 f_3(\alpha\textit{-status}(s)) + \lambda_4 f_4(a) \\ &\quad + \textstyle\sum_{e \in E} EQ_e^d \cdot e\textit{-response}(s) \end{aligned} \tag{4}$$

The f_1 function reflects the inherent value of performing a role as the team originally expected, hence deterring the agent from coordination changes (separate from the cost of the coordination change itself). The f_2 function reflects the value of keeping the agent's expectation of their performance of the role in agreement with the team's understanding of how the role will be performed. The overall reward is reduced based on the magnitude of the difference between the expectation and the reality. That is, the agent receives most reward when the role is performed exactly as the team expects, thus encouraging it to keep other team members informed of the role's status. The third component of the reward function, f_3, heavily influences overall reward based on the successful completion of the joint activity (which is after all the goal). This component encourages the agent to take actions that lead to the joint activity succeeding. The fourth component, f_4, factors in the cost and benefits of action and varies with the type of action and can be broken down further as follows:

$$f_4(a) = \begin{cases} \mathcal{D}_{cost} & \text{if } a = \mathcal{D} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

This component discourages the agent from taking costly actions (like coordination changes) unless it can gain some indirect value from doing so. The final component captures the value of getting a response from a decision-making entity.

Given the MDP's state space, actions, transition probabilities, and reward function, an agent can use *value iteration* to generate a policy $P{:}S \rightarrow \Gamma$ that specifies the optimal action in each state (Puterman 1994). The agent then executes the policy by taking the action that the policy dictates in each and every state in which it finds itself. A policy may include several transfers of control and deadline-delaying actions, as

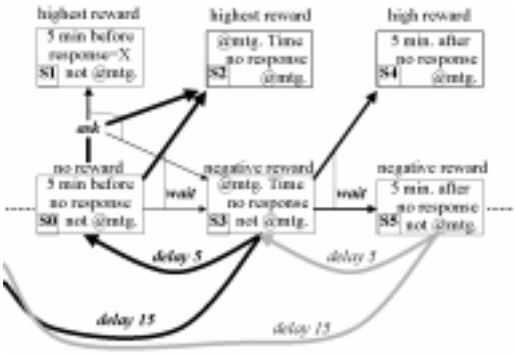


Figure 4: A small portion of the *delay MDP*.

well as a final autonomous action. The particular series of actions depends on the activities of the user.

Example: Delay MDP

One example of such an AA MDP is the *delay MDP*, covering all meetings for which Friday may act on behalf of its user. We model the particular AA decision within our general framework as follows. The joint activity, α, is for the meeting attendees to attend the meeting simulataneously. The agent's role, ρ, is to ensure that its user arrives at the currently scheduled meeting time. The constraints between the agent's role and the roles of other agents is that they occur simultaneously (i.e., the users must attend at the currently scheduled time). Changing the coordination of α corresponds to delaying the meeting. Friday has a variety of $\mathcal{D}$ actions at its disposal, including delays of various lengths, as well as cancellation of the meeting entirely. The user can also request a coordination change, e.g., via the dialog box in Figure , to buy more time to make it to the meeting. If the user decides a coordination change is required, Friday is the conduit through which other Fridays (and hence their users) are informed.

In the delay MDP's state representation, *team-orig-expect-ρ* is *originally-scheduled-meeting-time*, since attendance at the originally scheduled meeting time is what the team originally expects of the user and is the best possible outcome. *team-expect-ρ* is *time-relative-to-meeting*, which may increase if the meeting is delayed. *α-status* becomes *status-of-meeting*. *user-expect-ρ* is not represented explicitly; instead, *user-location* is used as an observable heuristic of when the user is likely to attend the meeting. For example, a user who is away from the deparment shortly before a meeting should begin is unlikely to be attending on time, if at all. Figure shows a portion of the state space, showing the *user-response*, and *user location* features. The figure also shows some state transitions (a transition labeled "delay n" corresponds to the action "delay by n minutes"). Each state contains other features (e.g., *previous-delays*), not pictured, relevant to the overall joint activity, for a total of 2760 states in the MDP for each individual meeting.

The general reward function is mapped to the *delay MDP* reward function in the following way. One component, denoted r_{time}, focuses on the user attending the meeting at the meeting time. r_{time} is the component of the reward modelling the difference between *team-expect-$\rho(s)$*, and *user-expect-$\rho(s)$*, i.e., the difference between what the team expected — arrive on time, and what the user did — arrive late. r_{time} is negative in states after the (re-) scheduled start of the meeting if the user is absent, but positive otherwise. The costs of changing the meeting time, i.e., the difference between *team-orig-expect-$\rho(s)$* and *team-expect-$\rho(s)$* is captured with r_{repair} and is proportional to the number of meeting attendees and the number and size of the delays. The final component, r_{user} captures the value of having the user at the meeting and is only received if the meeting actually goes ahead. r_{user} corresponds to α-$status$ in the general reward function. r_{user} gives the agent incentive to delay meetings when its user's late arrival is possible, but large delays incur a team cost from rearranging schedules.The overall reward function for a state, *s*, is a weighted sum: $r(s) = \lambda_{user} r_{user}(s) + \lambda_{repair} r_{repair}(s) + \lambda_{time} r_{time}(s)$.

The delay MDP's transition probabilities represent the likelihood that a user movement (e.g., from office to meeting location) will occur in a given time interval. Figure shows multiple transitions due to "ask" (i.e., transfer control to the user) and "wait" actions, with the thickness of the arrows reflecting their relative probability. The designer encodes the initial probabilities, which a learning algorithm may then tailor to individual users. Other state transitions correspond to uncertainty associated with a user's response (e.g., when the agent performs the "ask" action, the user may respond with specific information or may not respond at all, leaving the agent to effectively "wait"). One possible policy produced by the delay MDP, for a subclass of meetings, specifies "ask" in state **S0** of Figure (i.e., the agent gives up some autonomy). If the world reaches state **S3**, the policy specifies "wait". However, if the agent then reaches state **S5**, the policy chooses "delay 15", which the agent then executes autonomously. Using our language of strategies, we can denote this policy as HDA.

Data from Real-World Use

The E-Elves was heavily used between June 1, 2000 and March, 2001 and by a smaller group of users since then. The agents run continuously, around the clock, seven days a week. The user base has changed over the period of execution, with usually five to ten proxy agents running for individual users, a capability matcher (with proxy), and an interest matcher (with proxy). Often, temporary Friday agents operate on behalf of special guests or other short-term visitors.

The most emphatic evidence of the success of the MDP approach is that, since replacing the C4.5 implementation, the agents have *never* repeated any of the catastrophic mistakes enumerated in Section . For instance, the agents do not commit error 4 from Section , because the domain knowledge encoded in the bid-for-role MDP specifies a very high cost for erroneously volunteering the user for a presentation. Thus, the generated policy never autonomously volunteers the user. Likewise, the agents never committed errors 1 or 2. In the delay MDP, the lookahead inherent in the policy gen-

eration allowed the agents to identify the future rewards possible through "delay" (even though some delays had a higher direct cost than that of "cancel"). The MDP's lookahead capability also prevents the agents from committing error 3, since they can see that making one large delay is preferable, in the long run, to potentially executing several small delays. Although the current agents do occasionally make mistakes, these errors are typically on the order of asking the user for input a few minutes earlier than may be necessary, etc. Thus, the agents' decisions have been reasonable, though not always optimal, although, the inherent subjectivity in user feedback makes a determination of optimality difficult.

The general effectiveness of E-Elves is shown by several observations. Since the E-Elves deployment, the group members have exchanged very few email messages to announce meeting delays. Instead, Fridays autonomously inform users of delays, thus reducing the overhead of waiting for delayed members. Second, the overhead of sending emails to recruit and announce a presenter for research meetings is now assumed by agent run auctions. Third, a web page, where Friday agents post their user's location, is commonly used to avoid the overhead of trying to track users down manually. Fourth, mobile devices keep us informed remotely of changes in our schedules, while also enabling us to remotely delay meetings, volunteer for presentations, order meals, etc. We have begun relying on Friday 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!!"*.

Figure 5a illustrates the number of meetings monitored for each user. Over the course of three months (June 1 to August 31) over 400 meetings where monitored. Some users had fewer than 20 meetings, while others had over 150. Most users had about 20% of their meetings delayed. Figure 5b shows that usually 50% or more of delayed meetings were autonomously delayed. In particular, in this graph, repeated delays of a single meeting are counted only once, and yet, the graphs show that the agents are acting autonomously in a large number of instances. Equally importantly, humans are also often intervening, indicating the critical importance of AA in Friday agents.

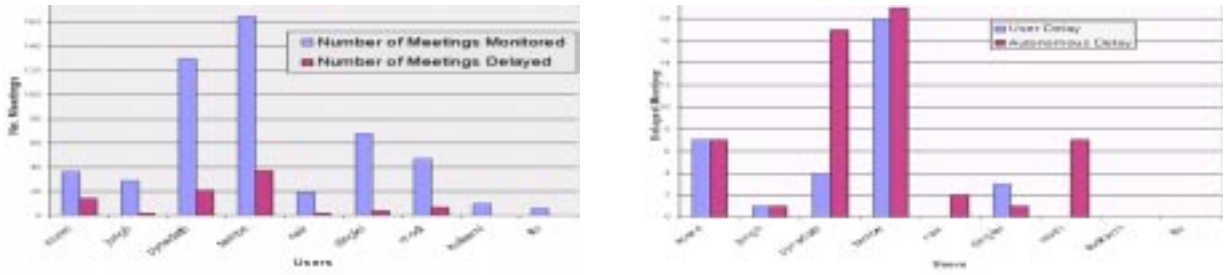
MDP Experiments

Experience using the MDP approach to AA in the E-Elves indicates that it is effective at consistently making reasonable AA decisions. However, in order to determine whether the MDP is a generally useful tool for AA reasoning, more conventional experiments are required. The reward function is engineered to encourage the reasoning to work in a particular way, e.g., the inclusion of a penalty for deviating from the original team plan should discourage the agent from asking for coordination changes unnecessarily. In this section experiments, designed to investigate the relationship between the reward function parameters and resulting polices, are presented. The first thing the experiments aim to do is verify that the policies change in the desired way when parameters in the reward function are changed. Secondly, from a practical perspective it is critical to understand how sensitive the MDP policies are to small variations in parameters because if the MDP is too sensitive to small variations it will be too difficult to deploy in practice. Finally, the experiments expose some unanticipated phenomena.

In each of the experiments we vary one of the λ parameters, i.e., the weights of the different factors, from Equation 5. The MDP is instantiated with each of a range of values for the parameter and many policies are produced. In each case the total policy has 2800 states. The policy is analyzed statistically to determine some basic properties of that policy, e.g., how many states the policy specifies to ask, to delay, etc. Such statistics give a broad feel for how the agent will act and highlights important characteristics of its approach. Notice that the percentage of each action that will actually be performed by the agent will not be the same as the percentage of times the action appears in the policy, since the agent will find itself in some states much more than in others. Hence, the statistics show *qualitatively* how the policy changes, e.g., asking more, rather then quantitatively how it changes, e.g., ask 3% more often.

The first experiment looks at the effect of the λ_1 parameter from Equation 5 on the policies produced by the delay MDP. This parameter determines how averse the agent should be to changing team plans. The parameter is primarily represented in the delay MDP by the *team repair cost* parameter. Figure 6 shows some properties of the policy change as the team repair cost value is varied. As the cost of delaying the meeting increases the agent will delay the meeting less (Figure 6(b)) and say not attending more often (Figure 6(d)). By doing this the agent gives the user less time to arrive at the meeting, choosing instead to just announce that the user is not attending. This is precisely the type of behavior that is expected, since it reduces the amount of time team mates will sit around waiting. The graph of the number of asks (Figure 6(a)) exhibits an interesting phenomena. For low values of the parameter the number of places in the policy where the agent will ask increases but for high values it decreases. For the low values, the agent can confidently make coordination changes autonomously, since their cost is low, hence there is less value to relinquishing autonomy. For very high coordination costs the agent can confidently decide autonomously not to make a coordination change. It is in the intermediate region that the agent is uncertain and needs to call on the user's decision making more often. The MDP in use in the E-Elves has this parameter set at 2. Around this value the policy changes little, hence slight changes in the parameter do not lead to large changes in the policy.

In the second experiment the λ_2 parameter is varied. This is the factor that determines how heavily the agent should weigh differences between how the team expects they will fulfill their role and how they will actually fulfill the role. In E-Elves this is primarily represented by the *team wait cost* parameter which determines the cost of having other team members waiting in the meeting room for the user. Figure 7 shows the changes to the policy when this parameter is varied. The graphs show that as the cost of teammates time increases the agent asks the user for input less often (Figure



(a) Monitored vs. delayed meetings per user (b) Meetings delayed autonomously (darker bar) vs. by hand.

Figure 5: Results of delay MDP's decision-making.

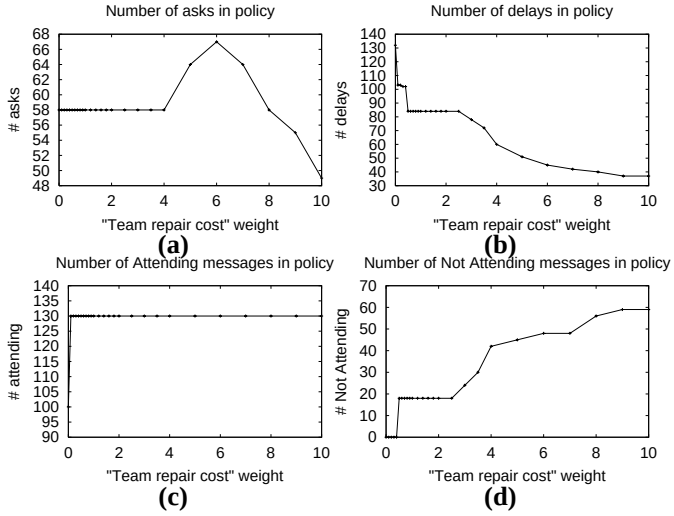


Figure 6: Properties of the MDP policy as team repair cost is varied.

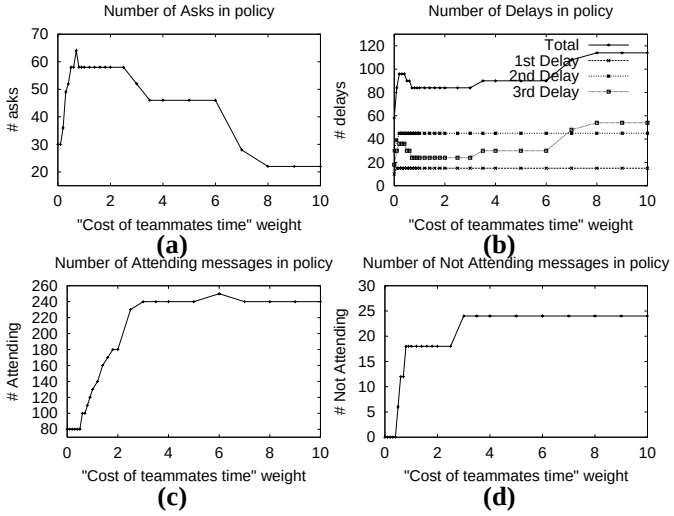


Figure 7: Properties of the MDP policy as team mate time cost is varied.

7(a)) and acts autonomously more often (Figure 7(b-d)). The agent asks whenever the potential costs of asking are higher than the potential costs of errors it makes. As the cost of time waiting for a user decision increases, the balance tips towards acting. Notice the phenomena of the number of asks increasing then decreasing occurs in the same way it did for the λ_1 parameter. In this case the agent acts when waiting costs are very low since the cost of its errors are very low, while when they are very high it acts because it cannot afford to wait for user input. In the E-Elves, a value of 1 is used for λ_2. This is in a relatively flat part of the graphs, indicating that detailed tuning of this parameter is not required. However, there is a reasonably significant change in the number of attending and not attending messages for relatively small changes in the parameter around this value, hence some tuning is required to get this to an appropriate setting.

The experiments show three important things. Firstly, changing the parameters of the reward function lead to the changes in the policy that are expected and desired. Second, the relatively smooth and predictable nature of most of the graphs indicates that detailed fine tuning of the parameters is not required to get the general characteristics policies qualitatively as desired. Finally, the interesting phenomena of the number of asks reaching a peak at intermediate values of the parameters was revealed.

Related Work

Several different approaches have been taken to the core problem of whether and when to transfer decision making control. While at least some of this reasoning is done in a team or multiagent context the possibility of multiple transfers of control is not considered. In fact, the possibility of delayed response leading to miscoordination does not appear to have been addressed at all. In the Dynamic Adaptive Autonomy framework a group of agents allocates a number of votes to each agent in a team, hence defining the amount of influence each agent has over a decision and thus, by their definition, the autonomy of the agent with respect to the goal(Barber, Martin, & Mckay 2000). The Dynamic Adapative Autonomy framework gives the team more detailed control over transfer of control decisions than does our approach, since only part of the decision making control can be transferred. However, since often more than one team

member will be able to vote on a decision, this approach is even more susceptible to miscoordination due to delayed response than was the failed C4.5 approach used in the E-Elves, yet there is no mechanism for flexible back and forth transfers of control.

Hexmoor(Hexmoor 2000) defines *situated autonomy* as an agent's *stance* towards a goal at a particular point in time. That stance is used to guide the agent's actions. One focus of the work is *how* an understanding of autonomy affects the agent's decision making at different decision making "frequencies", e.g., reflex actions and careful deliberation (Hexmoor 1999). For example, for reactive actions only the agent's *pre-disposition* for autonomy towards the goal is taken, while for decisions with more time available a detailed assessment is done to optimize the autonomy stance. Like our work, Hexmoor focuses on time as being an important factor in AA reasoning. However, the time scales looked at are quite different. Hexmoor looks at how much time is available for AA reasoning and decides which reasoning to do based on the amount of time available. We take the amount of time available into account while following the same reasoning process. Hence, Hexmoor's approach might be more appropriate for very highly time constrained environments, i.e., of the order of seconds.

Horvitz et al(Horvitz, Jacobs, & Hovel 1999) have looked at AA for reducing annoying interruptions caused by alerts from the variety of programs that might be running on a PC, e.g., notification of new mail, tips on better program usage, print jobs being finished and so on. Decision theory is used to decide, given a probability distribution over the user's possible focii of attention and potential costs and benefits of action, whether the agent should take some action autonomously. The agent has the further possibility of asking the user for information in order to reduce its decision making uncertainty. As described above, the reasoning balances the costs and benefits of autonomous action, inaction and a clarification dialog and then takes a one shot decision. Provided the interruptions are not critical the approach might be sufficient, however, if a team context was introduced, e.g., the incoming mail requiring notification is a request for an urgent meeting, our experiences with E-Elves suggest that a more flexible approach will be necessary.

While, to the best of our knowledge, E-Elves is the first deployed application using sophisticated AA reasoning there are reported prototype implementations of AA which demonstrate various ideas. Experiments using a simulated naval radar frequency sharing problem, show that different decision making arrangements lead to different performance levels depending on the particular task(Barber, Goel, & Martin 2000). This is an important result because it clearly shows AA can improve system performance by showing no one particular autonomy configuration is right for all situations.

An AA interface to the 3T architecture (Bonasso *et al.* 1997) has been implemented to solve human-machine interaction problems experienced using the architecture in a number of NASA projects (Brann, Thurman, & Mitchell 1996). The experiences showed that interaction with the system was required all the way from the deliberative layer through to detailed control of actuators. At the deliberative, planning level of 3T the user can influence the developed plan while the system ensures that hard constraints are not violated. At the middle level, i.e., the conditional sequencing layer, either the human user or system (usually a robot) can be responsible for the execution of each task. Each task has a pre-defined autonomy level that dictates whether the system should check with the user before starting on the action or just go ahead and act. The AA at the reactive level is implemented by a *tele-operation* skill that lets the user take over low level control, overriding commands of the system. The AA controls at all layers are encapsulated in what is referred to as the 3T's fourth layer – the interaction layer (Schreckenhost 1999). A similar area where AA technology is required is for safety critical intelligent software, such as for controlling nuclear power plants and oil refineries(Musliner & Krebsbach 1999). The work has resulted in a system called AEGIS (Abnormal Event Guidance and Information System) that combines human and agent capabilities for rapid reaction to emergencies in a petro-chemical refining plant. AEGIS features a *shared task representation* that both the users and the intelligent system can work with (Goldman *et al.* 1997). A key hypothesis of the work is that the model needs to have multiple levels of abstraction so the user can interact at the level they see fit. Both the user and the system can manipulate the shared task model. The model, in turn, dictates the behaviour of the intelligent system.

Meta-reasoning makes a choice of computations given the fact that completely rational choice is not possible(Russell & Wefald 1989). The idea is to treat computations as actions and "meta-reason" about the EU of doing certain combinations of computation and (base-level) actions. In general this is just as intractable as pure rationality at the "base-level" hence approximation techniques are needed at the meta level. AA can be viewed in essentially the same framework by viewing entities at computations. Then the AA meta-reasoning performs the same essential function as the meta-reasoning in an agent, i.e., choose computations to maximise EU. However, like much earlier AA work meta-reasoning does not consider the possibility of several transfers of control.

Conclusion

The E-Elves provides a unique opportunity for doing research into AA for complex multi-agent systems. Our early experiences dramatically demonstrated that single shot approaches from earlier work failed to meet the challenges of acting in cooperation with other agents. To avoid miscoordination, while not forcing an agent into risky decisions, we introduced the notion of a transfer of control strategy. An important aspect of the transfer of control strategies is the ability for the agent to change team coordination in order to buy more time for a decision to be made. Transfer of control strategies are operationalized via MDPs which creates a policy for the agent to follow. The MDP's reward function takes into account team factors, including the benefit of having the team knowing the status of individual roles and the cost of changing coordination. The MDP version of AA reasoning

used in the E-Elves performs well, not making the mistakes of the earlier, simpler implementation. Moreover, experiments show that the policies produced by the MDP exhibit a range of desirable properties, e.g., delaying activities less often, when the cost of doing so is high. The experiments indicate that MDPs are a practical and robust approach to implementing AA reasoning.

References

Barber, K.; Goel, A.; and Martin, C. 2000. Dynamic adaptive autonomy in multi-agent systems. *Journal of Experimental and Theoretical Artificial Intelligence* 12(2):129–148.

Barber, K. S.; Martin, C.; and Mckay, R. 2000. A communication protocol supporting dynamic autonomy agreements. In *Proceedings of PRICAI 2000 Workshop on Teams with Adjustable Autonomy*, 1–10.

Bonasso, R.; Firby, R.; Gat, E.; Kortenkamp, D.; Miller, D.; and Slack, M. 1997. Experiences with an architecture for intelligent reactive agents. *Journal of Experimental and Theorectical Artificial Intelligence* 9(1):237–256.

Brann, D.; Thurman, D.; and Mitchell, C. 1996. Human interaction with lights-out automation: A field study. In *Proceedings of the 1996 symposium on human interaction and complex systems*, 276–283.

Chalupsky, H.; Gil, Y.; Knoblock, C.; Lerman, K.; Oh, J.; Pynadath, D.; Russ, T.; and Tambe, M. 2001. Electric elves: Applying agent technology to support human organizations. In *International Conference on Innovative Applications of AI*, 51–58.

Collins, J.; Bilot, C.; Gini, M.; and Mobasher, B. 2000. Mixed-initiative decision-support in agent-based automated contracting. In *Proceedings of the International Conference on Autonomous Agents (Agents'2000)*.

Dorais, G.; Bonasso, R.; 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*, 397–420.

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.

for Papers, C. 1999. Aaai spring symposium on aa. www.aaai.org.

Goldman, R.; Guerlain, S.; Miller, C.; and Musliner, D. 1997. Integrated task representation for indirect interaction. In *Working Notes of the AAAI Spring Symposium on computational models for mixed initiative interaction*.

Gunderson, J., and Martin, W. 1999. Effects of uncertainty on variable autonomy in maintainance robots. In *Agents'99 workshop on autonomy control software*, 26–34.

Hexmoor, H. 1999. Adjusting autonomy by introspection. In *Proceedings of AAAI Spring Symposium on Agents with Adjustable Autonomy*, 61–64.

Hexmoor, H. 2000. A cognitive model of situated autonomy. In *Proceedings of PRICAI-2000, Workshop on Teams with Adjustable Autonomy*, 11–20.

Horvitz, E.; Jacobs, A.; and Hovel, D. 1999. Attention-sensitive alerting. In *Proceedings of UAI'99, Conference on Uncertainty and Artificial Intelligence*, 305–313.

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

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.

Musliner, D., and Krebsbach, K. 1999. Adjustable autonomy in procedural control for refineries. In *AAAI Spring Symposium on Agents with Adjustable Autonomy*, 81–87.

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

Pynadath, D. V.; Tambe, M.; Chalupsky, H.; Arens, Y.; et al. 2000. Electric elves: Immersing an agent organization in a human organization. In *Proceedings of the AAAI Fall Symposium on Socially Intelligent Agents*.

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

Russell, S. J., and Wefald, E. 1989. Principles of metareasoning. In Brachman, R. J.; Levesque, H. J.; and Reiter, R., eds., *KR'89: Principles of Knowledge Representation and Reasoning*. San Mateo, California: Morgan Kaufmann. 400–411.

Schreckenhost, D. 1999. Human interaction with control software supporting adjustable autonomy. In Musliner, D., and Pell, B., eds., *Agents with adjustable autonomy*, AAAI 1999 spring symposium series, 116–119.

Tambe, M.; Pynadath, D. V.; Chauvat, N.; Das, A.; and Kaminka, G. A. 2000. Adaptive agent integration architectures for heterogeneous team members. In *Proceedings of the International Conference on MultiAgent Systems*, 301–308.

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