Human Directability of Agents

Karen L. Myers David N. Morley
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
myers@ai.sri.com morley@ai.sri.com

Abstract

Many potential applications for agent technology require humans and agents to work together in order to achieve complex tasks effectively. In contrast, much of the work in the agents community to date has focused on technologies for fully autonomous agent systems. This paper presents a framework for the *directability* of agents, in which a human supervisor can define policies to influence agent activities at execution time. The framework focuses on the concepts of *adjustable autonomy* for agents (i.e., varying the degree to which agents make decisions without human intervention) and *strategy preference* (i.e., recommending how agents should accomplish assigned task). The directability framework has been implemented within a PRS environment, and applied to a multiagent intelligence-gathering domain.

Keywords

Advisable Systems, Agents, Mixed-initiative Control

INTRODUCTION

The technical and public press are filled these days with visions of a not-too-distant future in which humans rely on software and hardware agents to assist with problem solving in environments both physical (e.g., smart offices, smart homes) and virtual (e.g., the Internet). The notion of *delegation* plays a central role in these visions, with humans offloading responsibilities to agents that can perform activities in their place.

Successful delegation requires more than the assignment of tasks. A good manager generally provides directions to a subordinate so that tasks are performed to his or her liking. To ensure effectiveness, the manager will monitor the progress of the subordinates, occasionally interrupting to provide advice or to resolve problems.

The agents research community has, for the most part, focused on the mechanics of building autonomous agents and techniques for communication and coordination among agents. In contrast, little attention has been paid to supporting human interactions with agents of the type required for extended problem-solving sessions. Most agent frameworks fall at the extremes of the interaction spectrum, either assuming full automation by the agents with no means for user involvement, or requiring human intervention at each step along the way. Recently, however, there has been increased interest in agent systems designed specifically to support interaction with humans (e.g., [2, 3, 5, 16]).

We are developing a framework, called Taskable Reactive Agent Communities (TRAC), that supports directability of a team of agents by a human supervisor. Within TRAC, the human assigns tasks to agents along with guidance that imposes boundaries on agent behavior. By adding, deleting, or modifying guidance at execution time, the human can manage agent activity at a level of involvement that suits his or her needs. In essence, our approach can be viewed as form of process management technology that enables human control of agent communities.

A key issue in developing technology to support agent directability is determining the types of guidance to be provided. This paper focuses on guidance for *adjustable agent autonomy* and *strategy preferences*. Guidance for adjustable autonomy enables a supervisor to vary the degree to which agents can make decisions without human intervention. Guidance for strategy preferences constitutes recommendations on how agents should accomplish assigned tasks.

The main contributions of this paper are the characterization of these forms of guidance, presentation of a formal language for representing such guidance, the description of a semantic model for satisfaction of such guidance by an agent, and techniques for enforcing such guidance during agent operation.

Effective delegation and management by a human supervisor also requires visibility into ongoing agent operations. Although not described in this paper, the TRAC framework includes a capability for *customizable reporting* that enables a supervisor to tailor the amount, type, and frequency of information produced by agents to meet his evolving needs.

K-CAP'01, October 22-23, 2001, Victoria, British Columbia, Canada.

Details can be found in [12].

The paper begins with a description of our underlying model for agents. From there, we present an informal characterization of guidance for adjustable autonomy and strategy preferences. Next, we describe a multiagent system, called TIGER, which instantiates the TRAC approach to directability for the application of multiagent intelligence gathering in a simulated natural disaster scenario. We use TIGER to provide concrete examples of the directability concepts throughout the paper. Following this description, we present our representation for guidance and describe both our semantic model for guidance satisfaction and techniques for guidance enforcement. The paper concludes with a discussion of related work and directions for further research.

AGENT MODEL

We adopt a typical Belief-Desire-Intention (BDI) model of agency in the style of [14], whereby an agent undertakes actions to address its desires, relative to its current beliefs about the operating environment. BDI agents are so-called due to the three components of their "mental state": *beliefs* that the agent has about the state of the world, *desires* to be achieved, and *intentions* corresponding to actions that the agent has adopted to achieve its desires.

Each agent has a library of *plans* that define the range of activities that an agent can perform to respond to events or to achieve assigned tasks; our plan model is based on [17]. Plans are parameterized templates of activities that may require variable instantiations to apply to a particular situation. The *cue* of a plan specifies a stimulus that activates the plan, either a new goal or a change in the agent's beliefs. *Preconditions* associated with plans define gating constraints that must be satisfied in order for a plan to be applied. A plan is said to be *applicable* to a world change (e.g., new goal or belief change event) when the plan cue matches the stimulus, and the plan preconditions are satisfied in the current world state. The *body* of a plan specifies how to respond to the stimulus, in terms of actions to perform and subgoals to achieve.

An agent's plan library will generally contain a range of plans describing alternative responses to posted goals or events. Sets of these plans may be *operationally equivalent* (i.e., they share the same cue and preconditions) but differ in the approach that they embody. Some form of meta-control policy can be defined to select among such alternatives, should the need arise.

A BDI interpreter runs a continuous *sense-decide-act* loop. In each iteration the agent executes a single step of one of its intentions on the basis of its current beliefs about the state of the world. This entails performing actions, adopting new goals to achieve, updating its set of beliefs, and updating the set of current intentions. Within this paradigm, agents make three classes of decisions:

D1 *whether to respond to new goals and events*
D2 *how to select among multiple applicable plans*
D3 *how to select instantiations for plan variables*.

Our directability framework assumes that agents are capable of fully autonomous operation. More concretely, an agent's plan library covers the range of activities required to perform its assigned tasks. This assumption means that agents do not depend on the human supervisor to provide knowledge for task execution. Within this setting, guidance provides customization of agent behavior to suit the preferences of the human supervisor. In many applications, such guidance will enable superior performance, given that few plan libraries will reflect the full the experience, breadth of knowledge, and reasoning capabilities that a human supervisor can bring to the decision-making process.

TRAC FRAMEWORK FOR AGENT DIRECTABILITY

Our model of agent directability focuses on general and task-specific policies to influence the activities undertaken by agents in their execution of assigned tasks. In particular, we emphasize the areas of (a) adjustable levels of agent autonomy and (b) strategy preferences that describe approaches to be used by an individual agent in executing assigned tasks. Given the need to adjust to dynamic environments, these guidance policies can be defined and modified at any point during agent execution.

Adjustable Autonomy

We define the autonomy of an agent to be the extent to which it is allowed to make decisions (specifically, D1 – D3) on its own. In situations where activities are routine and decisions straightforward, a human may be content to delegate all problem-solving responsibility to an agent. However, in situations where missteps could have severe consequences, the degree of autonomy of an individual agent should necessarily be controllable by a human.

Because we are interested in domains where agents will need to operate with high degrees of autonomy, we assume a *permissive* environment: unless stated otherwise, agents are allowed to operate independent of human interaction. Our approach allows the human to adjust the scope of operations that can be undertaken by an agent on its own terms, focusing on the notions of *permission requirements* for action execution and *consultation requirements* for decision making.

Permission Requirements Permission requirements declare conditions under which an agent must elicit authorization from the human supervisor before executing actions. For example, the directive "Obtain permission before abandoning survey tasks with Priority $>$ 3" imposes the constraint that an agent request approval from its supervisor to abandon a certain class of tasks.

Consultation Requirements Consultation requirements designate a class of agent decisions that should be deferred to the human supervisor. These decisions relate to the selection of

values for variable instantiation, for example, "Consult when selecting locations for staging bases."

Our model of permission and consultation requirements, like earlier work on authority models, provides a mechanism to block performance of certain actions by an agent. However, authority models are generally static (e.g., the *levels of autonomy* in [2]) and often derived from organizational structures. In contrast, our approach provides a rich language for expressing permission and consultation policies, which can vary throughout a problem-solving session.

Strategy Preference

Strategy preferences express recommendations on how an agent should accomplish tasks. These preferences could indicate specific plans to employ or restrictions on plans that should not be employed, as well as constraints on how plan variables can be instantiated.

For example, the directive "Try contacting Nongovernmental Organizations for information before sending vehicles to towns on the west coast" expresses a preference for selecting among operationally equivalent plans. On the other hand, the directive "Only use helicopters for survey tasks in sectors that are expected to be inaccessible by truck for more than 1 week" restricts the choice of resource type for instantiating certain plan variables.

THE TIGER SYSTEM

We have developed a prototype implementation of our TRAC framework for agent guidance on top of the Procedural Reasoning System (PRS) [7]. The TRAC implementation has been used as the basis for a demonstration system called TIGER (TRAC Intelligence Gathering and Emergency Response) that serves as a testbed for exploring our ideas on agent directability. Within TIGER, a human supervisor can delegate tasks to agents while providing guidance to control their runtime behavior.

TIGER Functionality

TIGER provides control over a collection of simulated physical assets (trucks and aircraft), each embodied as a separate agent. These physical assets can be tasked to perform a range of actions related to intelligence gathering, and to provide assistance with eventualities such as medical emergencies, evacuations, and infrastructure repair.

TIGER serves as part of a *disaster response team* whose objective is to provide humanitarian relief in the wake of a natural disaster. Other organizations within the team provide logistics (e.g., supplies distribution), operations (e.g., repair of infrastructure), and medical services. These organizations have their own physical assets (trucks and aircraft) available for their use. As would be expected, these organizations need to share information and resources to perform their functions effectively. A human commander oversees operations, dynamically tasking organizations to implement the relief process.[1]

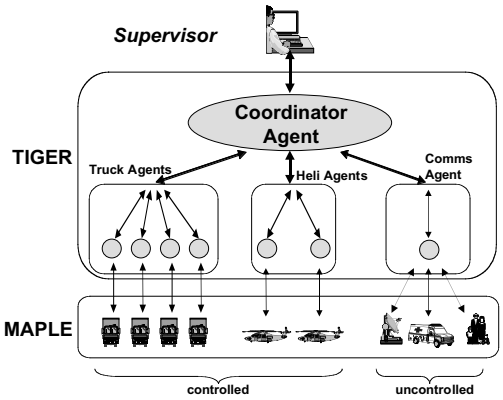

Figure 1. TIGER Architecture

The primary role for TIGER is to gather information in response to requests initiated by other members of the disaster response team or the supervisor. These requests can result in tasks to seek out information such as the current state of infrastructure (roads, bridges) in designated regions, or to collect supply requirements (medical, food, water, shelter) of designated population centers within impacted regions. There can also be requests to be informed of key events (such as medical emergencies) as they become known.

A secondary role is to respond to certain types of unexpected events (e.g., participating in evacuations, assisting with medical emergencies). Thus, TIGER agents must incorporate reactive capabilities that balance responsiveness with ongoing goal attainment.

The scope and complexity of the intelligence-gathering operations within the disaster relief context preclude step-by-step management of agent operations by a human commander. However, effective coordination of the available assets requires human supervision. As such, this domain provides an excellent example of an application that will benefit from technology for agent directability.

Agent Community Organization

Figure 1 displays the organization of agents within TIGER. The system has at its disposal a collection of simulated *physical agents* (trucks and helicopters) that can be used to gather information and respond to emergencies. In addition, there is a set of simulated *communications agents* (other relief organizations, nongovernment organizations, local officials) that can be consulted to obtain information. TIGER contains a

[1] The system operates within a testbed that simulates a major hurricane in Central America; the testbed is built on the MAPLE system (`http://www.cs.cmu.edu/~maple/`).

separate controller for each of the physical agents, as well as a communications manager for interacting with the various simulated communications agents. We refer to these controller agents as the *task execution agents* within TIGER, because they instigate and manage the activities required to perform assigned tasks.

The *coordinator agent* provides global management of tasks within the community, acting as a mediator between the human supervisor and the task execution agents. It also manages interactions with members of the disaster response team who request information (i.e., its *information clients*).

Tasking Model

The TIGER coordinator agent places incoming task requests into a pool of waiting tasks. It also maintains a pool of currently unallocated agents. The coordinator agent matches a waiting task with an unallocated agent based on properties of the task, the available agents, and current knowledge about the state of the roads and bridges. Task properties include *location*, *priority* (an integer from 0 to 10), *type* (e.g., survey, rescue), *deadline* (for completing the task), and *status* (e.g., pending, completed, failed). The agent properties include *agent type* (helicopter or truck) and *location*.

Task management constitutes a major component of an execution agent's decision-making process. An execution agent must determine what to do if, while executing one task, the coordinator agent passes it a second task. It must also decide when to drop tasks that are not progressing well in favor of new tasks with higher potential for success.

For simplicity, we limit each task execution agent to at most one active task at any point in time. Agents may also have pending tasks (which they intend to undertake) and preempted tasks (which were begun but put aside for higher-priority tasks). Tasks are assigned to individual agents and do not require coordination with other agents for their completion.

Unexpected events, such as a medical emergency, may require immediate response. Events are characterized by the properties *location*, *time* (of the event), *severity* (an integer 0 to 10), *number of people affected*, and *type* (e.g., evacuation, medical). Rather than creating a new task for the task pool, the coordinator agent selects an appropriate task execution agent to deal directly with each event.

These characteristics of tasking simplify the decision process for what an execution agent should do when it receives a task request. The agent can choose among several combinations of actions, including *ignore* the event, *adopt* a new task to respond to the event, *abandon* the current active task, *transfer* the task to another agent, or *postpone* the current task until the new task is completed. Alternatives in the agent's plan library encode each of these choices.

REPRESENTATION OF GUIDANCE

Our language for representing agent guidance builds on three main concepts: the underlying *agent domain theory*, a *domain metatheory*, and the connectives of first-order logic. Using these elements, we develop the main concepts underlying our model of agent guidance. These consist of an *activity specification* for describing abstract classes of action, a *desire specification* for describing abstract classes of goals, and an *agent context* for describing situations in which guidance applies.

Domain Metatheory

A standard domain theory for an agent consists of four types of basic element: *individuals* corresponding to real or abstract objects in the domain, *relations* that describe characteristics of the world, *goals* that an agent may adopt, and *plans* that describe available means for achieving goals.

The *domain metatheory* provides an abstracted characterization of elements of the domain theory that highlights key semantic differences. As discussed in [11], a metatheory can yield a rich vocabulary for describing activity, thus providing a powerful basis for supporting user communication. The main concepts within our metatheory for agent guidance are *features* and *roles* (similar in spirit to those of [10]) defined for agent plans and goals.

Consider first plans. A *plan feature* designates an attribute of interest for a plan that distinguishes it from other plans that could be applied to the same task. For example, among plans for route determination, there may be one that is `OPTIMAL` but `SLOW` with a second that is `HEURISTIC` but `FAST`; each of these attributes could be modeled as a feature. Although the two plans are operationally equivalent (i.e., same cue and preconditions), their intrinsic characteristics differ significantly. Features provide the means to distinguish between such operationally equivalent alternatives.

A *plan role* describes a capacity in which a domain object is used within a plan; it maps to an individual variable within a plan. For instance, a route determination plan may contain variables `location.1` and `location.2`, with the former corresponding to the `START` and the latter the `DESTINATION`.

In analogous fashion, roles and features can also be defined for goals. For example, a goal of informing another party of task progress may have a `COMMUNICATION` feature and `RECIPIENT` role associated with it. These metatheoretic constructs can be used to specify the class of goals that involve communicating with the commander.

Activity Specification

An *activity specification* characterizes an abstract class of plan instances for an agent. Our domain metatheory provides the basis for defining an activity specification, in terms of a set of required and prohibited features on a plan, as well as constraints on the way in which plan roles are filled.

Definition 1 (Activity Specification) *An* activity specification $\alpha = \langle \mathcal{F}^+, \mathcal{F}^-, \mathcal{R}, \phi \rangle$ *consists of*

- *a set of* required features $\mathcal{F}^+$
- *a set of* prohibited features $\mathcal{F}^-$
- *a set of* roles $\mathcal{R} = [R_1, \ldots, R_k]$ *and*
- *a* role-constraint *formula* $\phi[R_1, \ldots, R_k]$

For example, the following activity specification describes the class of plan instances with the feature `SURVEY` but not `HEURISTIC`, where the variables that fill the roles `START` and `DESTINATION` are instantiated to values in the same sector.

```
<{SURVEY},{HEURISTIC},
 {START, DESTINATION},
 {(= (SECTOR START) (SECTOR DESTINATION))}>
```

Desire Specification

A *desire specification* constitutes the goal-oriented analogue of an activity specification, consisting of a collection of required features, prohibited features, roles, and role constraints for goals. We use the symbol δ to represent a generic desire specification.

Agent Context

Just as individual plans employ preconditions to limit their applicability, guidance requires a similar mechanism for defining scope. To this end, we introduce the notion of an *agent context*. While plan preconditions are generally limited to beliefs about the world state, our model of agent context focuses on the full operational state of an agent, characterized in terms of its beliefs, desires, and intentions. Beliefs are specified in terms of constraints on the current world state. Desires are specified as desire specifications that describe goals that the agent has adopted. Intentions are specified through activity specifications that describe plans currently in execution by the agent.

Our model of agency assumes a hierarchical collection of plans and goals; furthermore, agents are capable of multitasking (i.e., addressing multiple goals in parallel). Within a given phase of the BDI execution cycle, goals for an agent of this type can be scoped in three different ways:

- *Current goal:* the goal for which the BDI interpreter is selecting a plan to execute
- *Local goals:* the current goal, or any of its ancestors
- *Global goals:* any goal of the agent

By distinguishing these different scopes for goals, guidance can be localized to more specific situations. Plans being executed can be scoped in a similar fashion.

Definition 2 (Agent Context) *An* agent context *is defined by a tuple* $\kappa = \langle \Phi, \Delta, A \rangle$*, where*

- Φ *is a set of well-formed formulae*
- $\Delta = \Delta^C \cup \Delta^L \cup \Delta^G$ *is a set of current, local, and global desire specifications, respectively*
- $A = A^L \cup A^G$ *is a set of local and global activity specifications, respectively.*[2]

Permission Requirements

Permission requirements are defined in terms of an *agent context* and a *permission-constrained activity specification*. The agent context defines conditions on the operating state of the agent that limit the scope of the permission requirement. The permission-constrained activity specification designates a class of plan instances for which permission must be obtained.

Definition 3 (Permission Requirement) *A* permission requirement $\langle \kappa, \alpha \rangle$ *consists of an agent context* κ *and an activity specification* α*.*

The interpretation of a permission requirement is that, when an agent's BDI state matches the specified agent context, permission must be obtained from the supervisor in order to execute a plan instance that matches the permission-constrained activity.

Example 1 (Permission Requirement) The statement "Seek permission to abandon survey tasks with priority > 5" could be translated into a permission requirement of the form

```
Agent Context:
 Local Activity Spec:
  Features+: SURVEY-TASK
Permission-Constrained Activity Spec:
  Features+: ABANDON
  Roles: CURRENT-TASK
  Constraint: (> (TASK-PRIORITY CURRENT-TASK) 5)
```

Consultation Requirements

A consultation requirements consists of an *agent context* and a *consultation role*. The interpretation of a consultation requirement is that when an agent's BDI state matches the agent context, any instantiation decision for a variable corresponding to the consultation role should be passed to the human supervisor.

Definition 4 (Consultation Requirement) *A* consultation requirement $\langle \kappa, R \rangle$ *consists of an agent context* κ *and a role R.*

Example 2 (Consultation Requirement) The guidance "When responding to medical emergencies, consult when selecting a medical evacuation site" would be translated into a permission requirement of the form

```
Agent Context:
 Local Activity Spec:
  Features+: EMERGENCY-RESPONSE, MEDICAL
Consultation Role: MEDEVAC-SITE
```

[2] Because the motivation for guidance is to influence the choice of plan for the current goal, we exclude the specification of a current plan from the intentions of an agent context.

Strategy Preference

Strategy preference guidance consists of two components: an *agent context* and a *response activity specification*. The activity specification designates the class of recommended plan instances to be applied (i.e., choice of plan and variable instantiations for designated roles) when the agent enters a state that matches the designated agent context.

Definition 5 (Strategy Preference) *A* strategy preference *rule is defined by a pair* $\langle \kappa, \alpha \rangle$ *where* κ *is an agent context and* α *is an activity specification.*

Example 3 The statement "Don't take on medical emergencies involving fewer than 5 people when the current task priority exceeds the emergency severity" could be represented by the following strategy preference:

```
Agent Context:
 Current Desire Spec:
  Features+: RESPOND-TO-EMERGENCY
  Roles: EVENT
  Constraint:
   (AND
    (= (EVENT-TYPE EVENT) MEDICAL)
    (< (EVENT-NUMBER-AFFECTED EVENT) 5))
Response Activity Spec:
  Features-: ADOPT
  Roles: EVENT, CURRENT-TASK
  Constraint:
    (> (TASK-PRIORITY CURRENT-TASK)
       (EVENT-SEVERITY EVENT))
```

A goal with the feature `RESPOND-TO-EMERGENCY` and role `EVENT` triggers consideration of the guidance, provided `EVENT` is a an emergency of type `MEDICAL`, and fewer than 5 people are affected. The response activity specification indicates not to adopt responsibility for the emergency in the event that the priority of `CURRENT-TASK` is greater than the severity of `EVENT`.

SEMANTICS AND ENFORCEMENT OF GUIDANCE

Space limitations preclude full descriptions of the formal semantics for satisfaction of guidance by agent execution and algorithms for guidance enforcement. We present a brief overview here; details can be found in [13].

Semantically, guidance acts as a filter on the plan instances that an agent can execute. When a standard BDI agent attempts to find an instance of a plan from its library to apply to a goal, it determines a set of applicable plan instances based on the plan cues and preconditions. The guidance limits this set further in accord with the following conventions.

A guidance rule is deemed *relevant* at the time that the applicable plans are being filtered if the agent context matches the current operational state of the agent. Each relevant strategy preference rule filters out plan instances that do not match the response activity specification. Each relevant permission requirement rule filters out plan instances that both match the permission-constrained activity specification and are refused permission by the supervisor. Each relevant consultation requirement rule filters out plan instances that have the consultation role but do not bind the corresponding role variable to a value desired by the supervisor.

Enforcement of guidance is attained through a simple modification to the standard BDI interpreter loop at the point where where a plan instance is selected in response to a posted goal. First, the current BDI operational state for an agent is matched to the agent context components of all currently defined guidance to determine the relevant guidance for the current execution cycle. The relevant strategy preference rules are then used to eliminate plan instances that do not match their response activity specification. The remaining plan instances are then ordered in accord with any meta-control policies for plan ordering that may have been defined. This list is then traversed in order to find the first for which either the plan instance is not affected by relevant permission or consultation requirement rules, or queries to the human supervisor elicit any required execution permissions and instantiations for role variables. The agent then applies the selected plan instance to the current goal.

CONFLICTING GUIDANCE

User guidance provides a powerful mechanism for runtime customization of agent behavior. However, it also introduces the potential for problems in the event that the guidance recommends inconsistent responses. Robustness of operations necessarily requires mechanisms that can detect problematic user guidance and respond in a manner that does not jeopardize the stability of an agent.

Conflicts can arise in different forms. Here, we distinguish between *direct* and *indirect* conflicts.

Direct conflicts arise when guidance yields recommendations that conflict with each other within a given cycle of the BDI interpreter. For example, *Execute plan P* and *Don't execute plan P*. Direct conflicts are easily detected. They can be resolved by associating *weights* with strategy preferences rules that indicate degree of preference. A policy for combining and comparing the weights associated with the strategy preference rules that made the conflicting recommendations can then be used to select a preferred response. TIGER incorporates an approach of this type to deal with direct conflicts.

Indirect conflicts arise when guidance recommends multiple plans for execution such that, while their execution can be initiated, it is impossible for all of them to complete successfully. For example, the simultaneous execution of two plans could lead to deadlock or livelock situations, or downstream resource contention. Powerful detection mechanisms are required to deal with this class of conflict; TIGER does not yet include capabilities of this type.

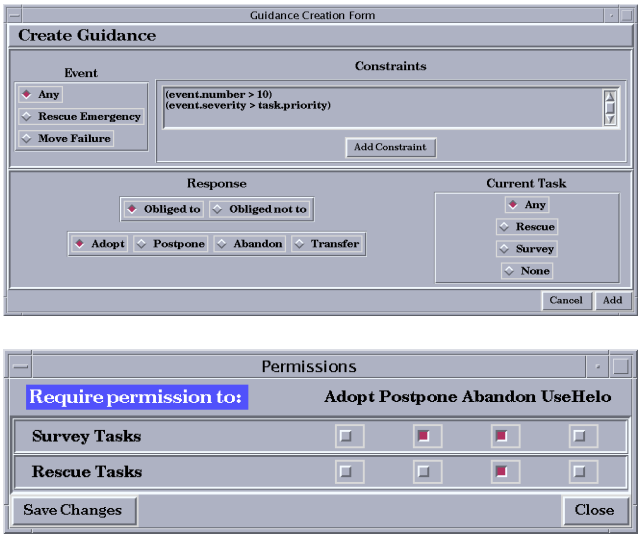

Figure 2. Selected Guidance Specification Tools

GUIDANCE INTERFACE TOOLS

The motivation for our work on agent directability is to enable users to more readily direct and manage agents in dynamic, unpredictable environments. The language presented in earlier sections provides a highly expressive formalism in which to define agent guidance; however, the complexity of the language could overwhelm a typical user. For this reason, we have been developing tools to help users define and manipulate agent guidance. Figure 2 presents two such tools from the TIGER system.

The first tool is a *guidance authoring interface* that walks the user through the process of constructing complex pieces of guidance. To enable a simple specification process, the tool does not support the full expressivity of the formal guidance language; however, it supports a broad range of expressions (including the examples described in this paper).

The second tool is a *permissions window*. It enables users to activate/deactivate permission requirements for certain classes of action performed on certain types of task. In particular, selections made through this interface are compiled into corresponding permission requirement structures. While this interface limits the scope of permission requirements that can be expressed, it provides a simple, accessible specification mechanism.

In addition to the two tools described above, we have also developed a *guidance library* that stores predefined pieces of guidance. Users can then select from predefined guidance, as appropriate, to address their needs in a particular situation.

RELATED WORK

Recognition of the need for technologies to support human-agent interactions has grown substantially in the past few years. To date, however, few concrete technical approaches have been proposed to address the problem of agent directability.

Scerri et al. [15] apply Markov decision processes (MDPs) to provide a form of adjustable autonomy. Their approach involves predefining an MDP for each agent to describe possible courses of action. The agent uses expected utility estimates from this model to determine when to consult the supervisor, and adjusts the model parameters based on experience. To avoid learning inappropriate behavior, users can define predefined constraints on what can be learned. In contrast to our approach of having a human explicitly define a policy for autonomy, an agent within this framework determines an appropriate level on its own.

Schreckenghost et al. [16] apply the concept of adjustable autonomy to the management of space-based life support systems. In their system, a human can take over both the selection of tasks to perform and the execution of those tasks. In contrast to our use of an explicit policy language, the level of autonomy is specified by directly altering a "level of autonomy" setting (*manual* versus *autonomous*) either for all tasks, for a subsystem, or for an individual task.

Our strategy preference guidance selects among previously defined alternative plans; it does not expand the behavioral capabilities of the agent. In contrast, the work on *policy-based control* for distributed systems managements supports the runtime definition of new behaviors (e.g., [9]). Policy languages in this community focus on the concepts of *authority* and *obligation* to perform actions.

CONCLUSION

This paper presents a framework for human directability of agents that enables a user to define polices for adjustable agent autonomy and strategy preference. Through these mechanisms, a human supervisor can customize the operation of agents to suit his individual preferences and the dynamics of unexpected situations. In this way, system reliability and user confidence can be increased substantially over fully autonomous agent systems. The power of these ideas has been demonstrated within the TIGER system, which supports a human intelligence officer in managing a community of agents engaged in tasks for information gathering and emergency response.

Many outstanding issues in this area remain to be addressed; we briefly describe three topics for future work.

Detecting and Resolving Guidance Conflicts As discussed above, TIGER recognizes only a limited class of guidance-related conflicts (namely, direct conflicts among guidance). Indirect conflicts among guidance, and conflicts between guidance and ongoing activities require more powerful detection methods that reason about the downstream effects and requirements of plans. We are also interested in expanding our simple prioritization approach to resolving direct con-

flicts among guidance to incorporate more advanced conflict resolution policies (e.g., [4, 8]).

Community Guidance The forms of agent directability described in this paper focus on influencing the behavior of an individual agent. Human supervisors will also want to express control at the *community* level, to encourage or discourage various forms of collective or emergent behaviors. The guidance Keep 2 trucks within 15 miles of headquarters. provides an example. Enforcement of this type of guidance will require mechanisms that support information exchange and coordinated action selection among groups of agents.

Collaborative Control Our model of agent directability provides a form of *supervised autonomy* [1] in which control over autonomy rests solely with the human supervisor. Some situations may benefit from a more collaborative approach [6], where both sides share control over initiative. For example, an agent may choose to initiate a dialogue with the human in situations where adherence to guidance would interfere with the pursuit of current goals, rather than blindly following the user's recommendations.

ACKNOWLEDGMENTS

The authors thank Eric Hsu for his contributions in developing the TIGER interface, and Sebastian Thrun and his group at CMU for providing the MAPLE simulator. This research was supported by DARPA Contract F30602-98-C-0160 under the supervision of Air Force Research Laboratory – Rome.

REFERENCES

1. K. S. Barber and C. E. Martin. Agent autonomy: Specification, measurement, and dynamic adjustment. In *Proceedings of the Autonomy Control Software Workshop at Autonomous Agents*, 1999.

2. P. Bonasso. Issues in providing adjustable autonomy in the 3T architecture. In *Proceedings of the AAAI Spring Symposium on Agents with Adjustable Autonomy*, 1999.

3. H. Chalupsky, Y. Gil, C. A. Knoblock, K. Lerman, J. Oh, D. Pynadath, T. A. Russ, and M. Tambe. Electric Elves: Applying agent technology to support human organizations. In *Proceedings of the Thirteenth Conference on Innovative Applications of Artificial Intelligence*, 2001.

4. F. Dignum, D. Morley, E. A. Sonenberg, and L. Cavedon. Towards socially sophisticated BDI agents. In *Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS'2000)*, 2000.

5. G. Ferguson and J. Allen. TRIPS: Towards a mixed-initiative planning assistant. In *Proceedings of the AIPS Workshop on Interactive and Collaborative Planning*, 1998.

6. T. Fong, C. Thorpe, and C. Baur. Collaborative control: A robot-centric model for vehicle transportation. In *Proceedings of the AAAI Spring Symposium on Agents with Adjustable Autonomy*, 1999.

7. M. P. Georgeff and F. F. Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

8. E. Lupu and M. Sloman. Conflicts in policy-based distributed systems. *IEEE Transactions on Software Engineering, Special Issue on Inconsistency Management*, 25(6), 1999.

9. J. D. Moffett and M. S. Sloman. Policy hierarchies for distributed systems management. *IEEE Journal on Selected Areas in Communications*, 11(9), 1993.

10. K. L. Myers. Strategic advice for hierarchical planners. In L. C. Aiello, J. Doyle, and S. C. Shapiro, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*. Morgan Kaufmann Publishers, 1996.

11. K. L. Myers. Domain metatheories: Enabling user-centric planning. In *Proceedings of the AAAI-2000 Workshop on Representational Issues for Real-World Planning Systems*, 2000.

12. K. L. Myers and D. N. Morley. Directing agent communities: An initial framework. In *Proceedings of the IJCAI Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*, 2001.

13. K. L. Myers and D. N. Morley. The TRAC framework for agent directability. Technical report, Artificial Intelligence Center, SRI International, 2001.

14. A. S. Rao and M. P. Georgeff. BDI agents: From theory to practice. In *Proceedings of the International Conference on Multi-Agent Systems (ICMAS-95)*, San Francisco, USA, 1995.

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

16. D. Schreckenghost, J. Malin, C. Thronesbery, G. Watts, and L. Fleming. Adjustable control autonomy for anomaly response in space-based life support systems. In *Proceedings of the IJCAI Workshop on Autonomy, Delegation, and Control: Interacting with Autonomous Agents*, 2001.

17. D. E. Wilkins and K. L. Myers. A common knowledge representation for plan generation and reactive execution. *Journal of Logic and Computation*, 5(6), 1995.