

User's Guide for the TIGER System

Version 1.2

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

April, 2002

1 Introduction

The technical press 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 off-loading responsibilities to agents that can perform activities in their place.

Successful delegation requires more than the mere 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 lie 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 (i.e., *teleoperation* mode). Recently, however, there has been increased interest in agent systems designed specifically to support interaction with humans (e.g., [1, 2, 3, 9]).

As part of the DARPA CoABS (Co-ordination of Agent Based Systems) program, SRI has developed a framework [6], called Taskable Reactive Agent Communities (TRAC), which 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.

A key issue in developing technology to support agent directability is determining the types of guidance to be provided. The TRAC framework supports 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. Effective delegation and management by a human supervisor also requires visibility into ongoing agent operations. 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.

The concepts of the TRAC framework for agent guidance have been implemented on top of the Procedural Reasoning System (PRS) [4]. 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 test bed for exploring these ideas on agent directability. TIGER plays the intelligence gathering role in a multi-agent disaster response simulation.

Within TIGER, a human supervisor can delegate tasks to agents while providing guidance to control their runtime behavior.

This User Guide describes the TIGER demonstration system, and how to install and run it. The Guide begins with a description of the TIGER system (Section 2). The next section provides instructions for installing TIGER on a Windows 2000 machine¹, and is followed by a section detailing how to start and interact with the TIGER system.

For a more detailed explanation of the TRAC framework, readers can consult the technical papers [5, 6, 8].

2 The TIGER System

2.1 TIGER Functionality

The TIGER system was created as a test bed for exploring the TRAC framework for agent guidance. It was developed as part of the MIATA demonstration system of the DARPA CoABS program.

The MIATA demonstration system brought together a number of diverse agent systems to serve as a *disaster response task force* whose objective is to provide humanitarian relief in the wake of a simulated natural disaster. TIGER played the intelligence gathering (J2) role in the task force. Other organizations within the task force provided logistics (e.g., supplies distribution), operations (e.g., repair of infrastructure), and medical services. Each organization had its own simulated physical assets (trucks and aircraft) available for its use. As would be expected, these organizations needed to share information and resources to perform their functions effectively. A human commander supervised operations, dynamically tasking organizations to implement the relief process.

Underlying the demonstration was a test bed that simulated a major hurricane in Central America; the test bed was built on the MAPLE system (<http://www.cs.cmu.edu/~maple/>).

The primary role for TIGER is to gather information in response to requests from the supervisor or other members of the disaster response team. These requests can result in tasks to acquire information on 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 unexpected events (e.g., participating in evacuations, assisting with medical emergencies). Thus, TIGER agents must incorporate reactive capabilities that balance

¹TIGER may run on other versions of the Windows operating system, but has not been tested.

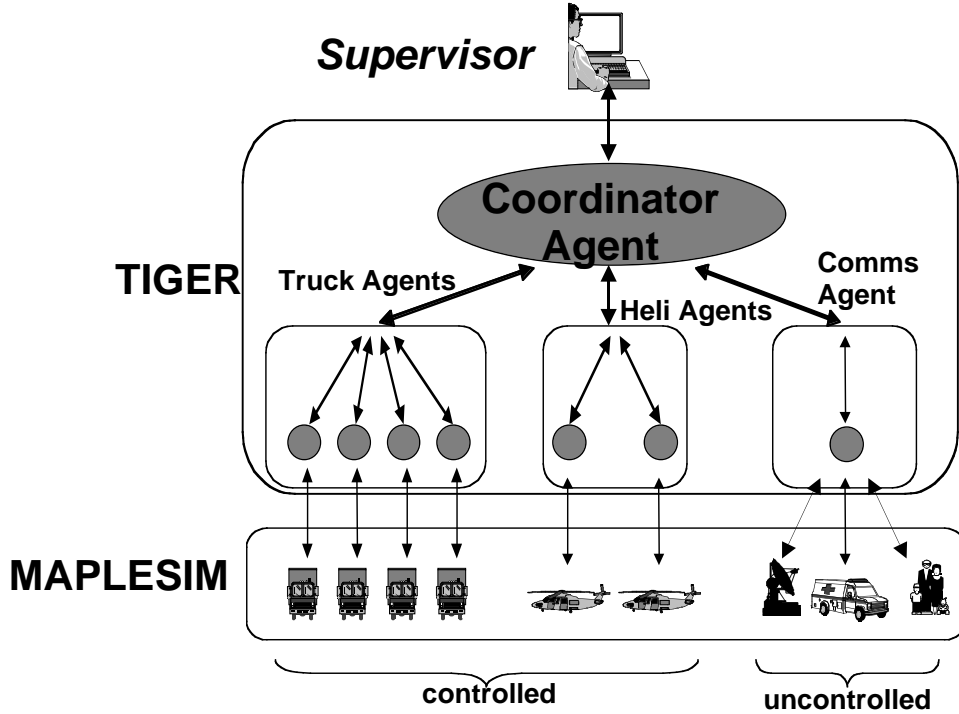


Figure 1: TIGER Architecture

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. 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.

2.2 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, nongovernmental organizations, local officials) that can be consulted to obtain information. TIGER contains a 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*).

2.3 Tasking Model

The TIGER coordinator agent maintains both a pool of unassigned tasks and 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), and *status* (e.g., pending, completed, failed). The agent properties include *type* (e.g., 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.

The task execution agents are responsible for handling each request from the coordinator agent to deal with a task or event. They must also deal with events caused by execution problems (for example, when a truck encounters a bridge so damaged as to be impassible).

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 (e.g., 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). The coordinator agent selects an appropriate task execution agent to deal directly with each such event, thus bypassing the task pool.

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. The agent's plan library includes options for each of these choices.

3 Installation of TIGER System

TIGER was designed to play the J2 role in the MIATA demonstration system. As such, it expected task requests and information to be supplied by other agents. TIGER can also be run in “stand-alone” mode, where the messages that the other joint task force members would normally supply are instead supplied by the user through the graphical user interface. This user guide documents the use of TIGER in stand-alone mode.

To run the TIGER demonstration system in stand-alone mode, three components are required:

- The TIGER executable and associated files – implements the TRAC framework and provides a graphical user interface for interacting with the system
- The CMU MAPLE simulator, which provides the simulated environment for TIGER to operate within and the SimSpy2 visualization tool for viewing the movement of simulated trucks and helicopters
- The Tcl/Tk language – used to implement the TIGER graphical user interface

TIGER – The TIGER executable and associated files are contained in a self-extracting archive called `TIGERINSTALL.exe`. Running this self-extracting archive results in a directory `TIGER`. This directory that does not need to be in any fixed location; however, the path to the directory cannot contain spaces. Thus, you should not put it under `C:/Documents and Settings/`, `C:/Program Files/`, on your desktop, and so on. One possible location is `C:/TIGER`.

CMU MAPLE simulator – The installer for the CMU MAPLE simulator and SimSpy2 can be downloaded from the “Simulation Tools v2.1” link on the web page: <http://www-2.cs.cmu.edu/~maple/simdemos.html>. When the downloaded file `simulator.exe` is run, it installs the simulator and SimSpy2.

Unfortunately, the executables in this installation are old. Newer executables are available and may be included in the `TIGER` directory; however, the installation step is necessary as the newer executables will not work without the installer having been run.

If your `TIGER` directory does not contain the CMU simulator and visualization tool (called `sim.exe` and `SimSpy2.exe`), you can download the latest versions from <http://www-2.cs.cmu.edu/~maple/newsim/> as files `Sim_NT.zip` and `SimSpy2.exe`. The `Sim_NT.zip` file expands to an executable called `sim.exe`. After downloading these files, move both `sim.exe` and `SimSpy2.exe` into your `TIGER` directory.

Tcl/Tk – If Tcl/Tk is not already installed on your machine, go to the ActiveState web site <http://aspn.activestate.com/ASPN/Tcl/Downloads/>, download ActiveTcl, and fol-

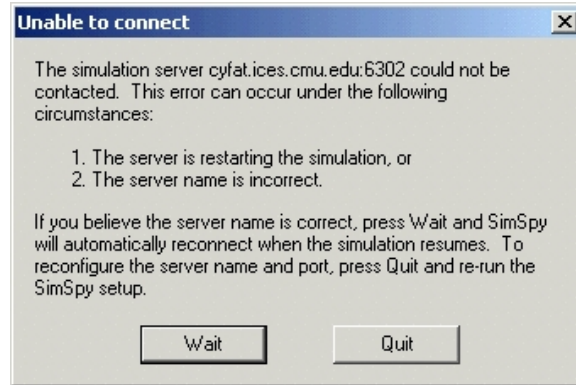


Figure 2: SimSpy2 Error Window

low the instructions to install Tcl/Tk in the standard location `C:/Tcl`. TIGER has been tested with version 8.3.4.2, the current version as of April 2002.

When initially installed SimSpy2 is set up to look for the simulator to be running on a machine at CMU. You need to change it to look for the simulator on the local machine (127.0.0.1) instead. This only has to be done once.

To make this change, start the simulator running by double-clicking on the “sim” executable in the TIGER directory. Once the simulator has started printing out its internal state (e.g., number of clients, agents, load), start SimSpy2 by double-clicking on the SimSpy2 executable in the TIGER directory. SimSpy2 will try to connect to a simulator running at CMU. Eventually, it times out with a message “Unable to connect” (see Fig. 2) and gives you the options “Wait” and “Quit”. Contrary to the instructions in the error message, select “Wait”. You will now have access to the SimSpy2 menus. Select the “Settings...” menu-item of the “Edit” menu. In the “SimSpy Settings” window that is presented, change the Server Host setting to 127.0.0.1 (see Fig. 3) and click “Save”, leaving the other fields untouched. Now quit out of SimSpy2 and quit the simulator by closing the simulator window.

4 Running the TIGER Demonstration System

The main steps for running the TIGER demonstration system are

- Start the components running: Maple simulator, TIGER and optionally SimSpy,
- Interact with TIGER:
 - Send messages to TIGER that it would normally receive from other MIATA agents

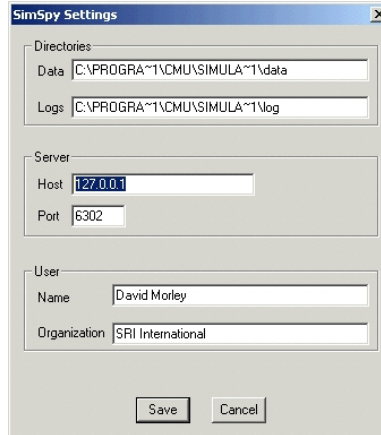


Figure 3: SimSpy2 Settings

- Set and manage the level of autonomy, reporting, and guidance for the TIGER agents
- Quit the components

4.1 Start-up

To start the simulator, double click on the “Sim” application in the TIGER directory. After an initialization period, the simulator will start printing out the state of the simulator, listing the number of connected client applications, the number of agents (trucks and helicopters), and so on.

At this point you can start up the SimSpy tool by double-clicking the SimSpy2 application in the TIGER directory.

To start TIGER, double-click on the TIGER application in the TIGER directory. This will run Allegro Common Lisp from the tiger.dxl image. To start the demo, type “:ld start” at the Lisp prompt. This loads the file “start.lisp” that performs some initializations, including connecting to the simulator, and then starts up a PRS interface (Figure 4). At this point, if you are familiar with PRS you can adjust the level of plan tracing desired. Now exit the PRS interface by selecting “Exit” from the “Application” menu in the top left of the PRS interface and the TIGER interface will start up. Three windows will appear: “Main Menu” (Figure 5), “Demo Events” (Figure 6), and “Reports Display” (Figure 7). Relocate these windows to convenient locations on your screen so you can see all three. You may need to resize the “Reports Display” window (NOTE: do not press any of the buttons on the interface until the PRS(2): prompt appears in the Lisp window).

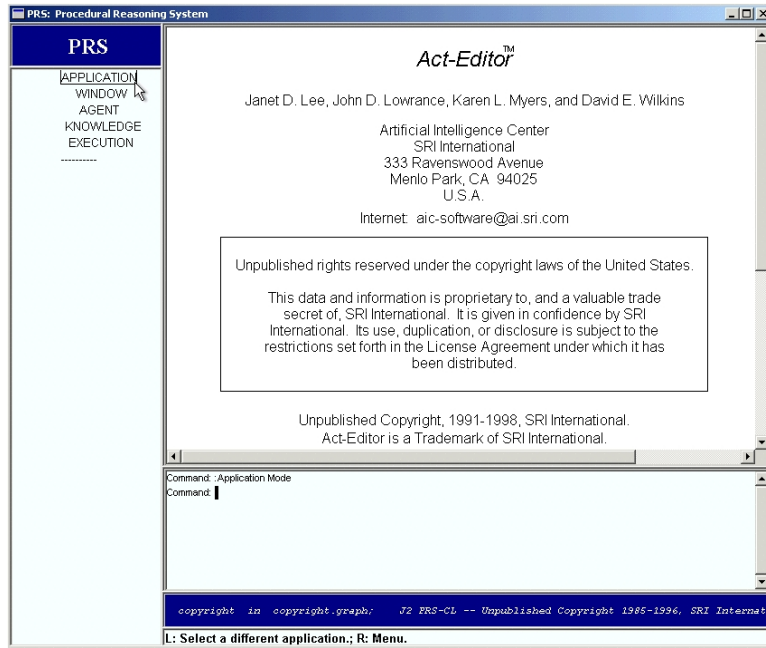


Figure 4: PRS Window



Figure 5: Main Menu

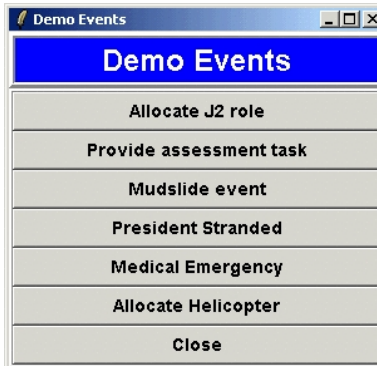


Figure 6: Demo Events

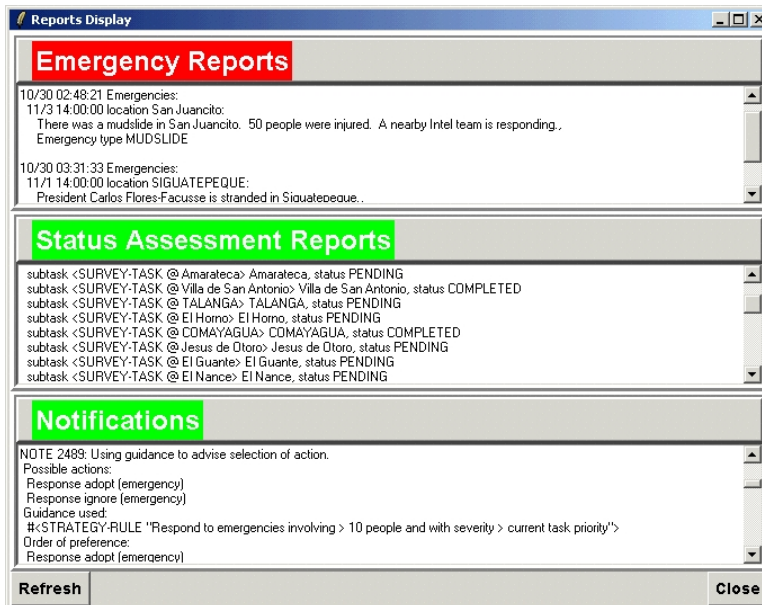


Figure 7: Reports Display

4.2 Interacting with TIGER – The Interface

The TIGER interface consists of three windows: “Main Menu”, “Demo Events”, and “Reports Display”. These three windows should not be closed while the demo is running.

The “Main Menu” window (Figure 5) represents the J2 commander’s main control interface for TIGER and contains seven buttons. These seven buttons are

- “Display Reports” – opens the “Reports Display” window (Figure 7) if it is not already open.
- “Set Permissions” – opens a “Permissions” window (Figure 8) where the level of autonomy of the agents can be set. This window contains an array of checkboxes for specifying whether permission should be sought when survey (assessment) or rescue tasks are to be adopted, postponed or abandoned, or when helicopters are to be used. When the “Save Changes” or “Close” button is pressed, any changes to the checkboxes are passed on to TIGER.
- “Set Guidance” – opens the “Guidance Activation Window” (Figure 9) where the J2 commander can create and activate guidance for the agents. This window allows the user to set guidance for the agents. The window lists currently known rules and next to each rule is a checkbox to specify whether or not the rule is active. The user can modify which rules are active by clicking in the checkboxes. When either the “Save Changes” or “Close” button is pressed, any changes to the checkboxes are passed on to TIGER. When TIGER is started up, it has a set of predefined guidance rules, with a subset of these marked active. However, the user can create additional rules by clicking on the “Add Guidance” button to bring up the “Guidance Creation Form” window described later in this section.
- “Customize Reporting” – opens a window (Figure 10) where the frequency and level of detail of agent reporting can be set.
- “Issue Commands” – opens the “Miscellaneous Commands” window (Figure 11). In this window the J2 commander can query the status of the tasks and send a request for additional helicopter resources from other MIATA agents.
- “(Demo Events)” – opens the “Demo Events” window if it is not already open. This window allows the user to supply TIGER with the messages that the other joint task force members would normally supply in the full MIATA demonstration system.
- “Quit” – quits the graphical user interface.

The “Demo Events” window (Figure 6) exists to allow the user to send messages to TIGER that other MIATA agents would normally supply in the full MIATA demonstration system. It contains seven buttons:



Figure 8: Permissions Window

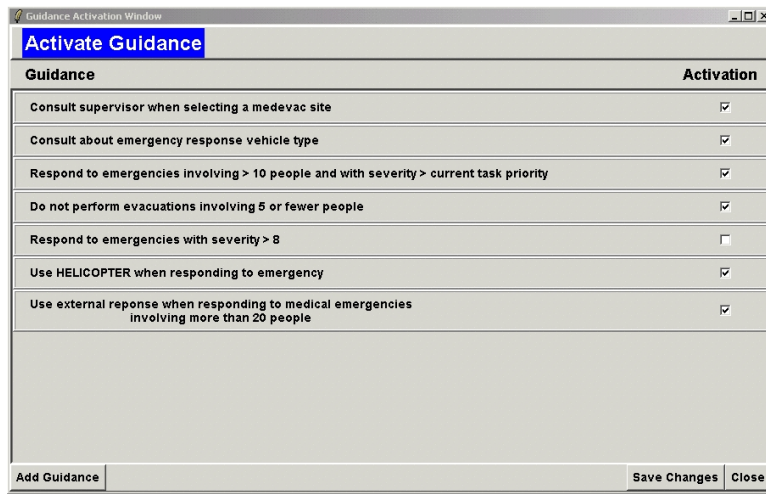


Figure 9: Guidance Activation Window

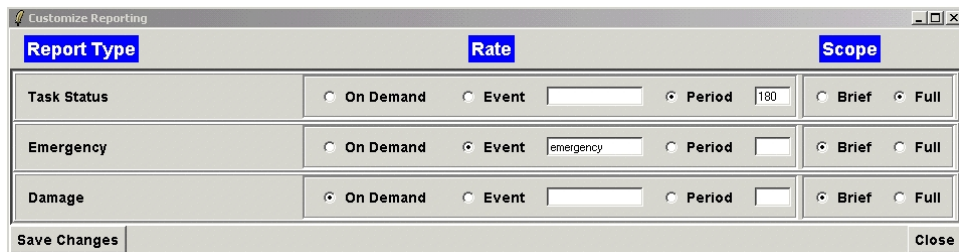


Figure 10: Customize Reporting Window

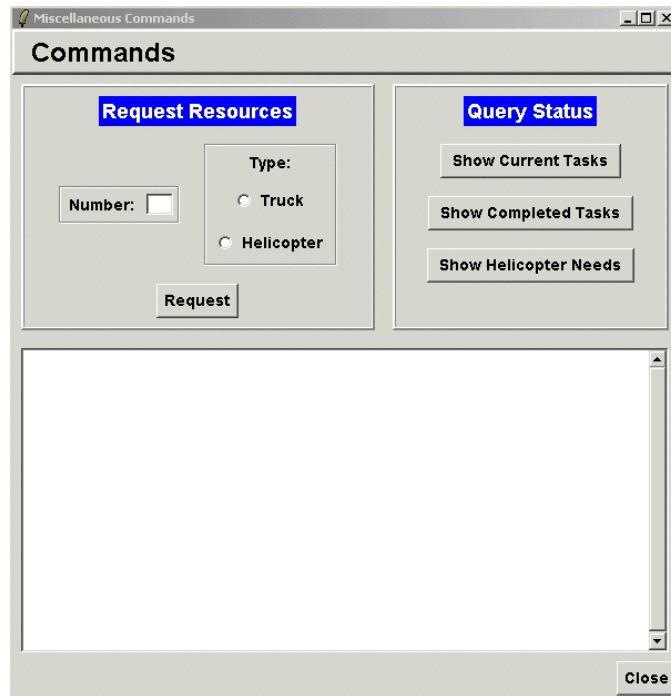


Figure 11: Miscellaneous Commands Window

- “Allocate J2 role” – the first message that should be sent to TIGER. It indicates that TIGER has been allocated the J2 role in the task force.
- “Provide assessment task” – requests the J2 to survey a specific area for damage. This is usually the second message to be sent to TIGER.
- “Mudslide event” – reports that there was a mudslide in San Juancito where 50 people have been injured. This information needs to be passed on to other interested agents, and the J2 will need to decide whether it should divert resources to assist.
- “President Stranded” – reports that the president has been stranded in Siguatepeque and needs evacuation.
- “Medical Emergency” – reports that there was a medical emergency in San Jose affecting 50 people.
- “Allocate Helicopter” – reports that the J2 has been allocated an additional helicopter.
- “Close” – closes the window.

The “Reports Display” window (Figure 7) contains three panes:

- The “Emergency Reports” pane lists emergency messages (the level of reporting is specified using the “Customize reporting” window).

Figure 12: Guidance Creation Form

- The “Status Assessment Reports” pane lists the progress of assessment tasks that have been allocated to the J2 (again, the level of reporting is specified using the “Customize reporting” window).
- The “Notifications” pane indicates the inner workings of TIGER. For example, when preferences among conflicting guidance rules come into play, it lists the rules that were used in deciding among possible courses of action.

The “Guidance Creation Form” window (Figure 12) allows the user to create new guidance rules on the fly. Here the user can specify

- The type of event to which the rule applies: a rescue emergency, a movement failure, or any
- Whether the type of the task currently being executed is relevant (rescue task, survey task, no task)
- The category of the response (adopt a new task, postpone an existing task, abandon an existing task, transfer an existing task) and whether the agent should engage or avoid that response
- Additional constraints on the applicability of the guidance

The guidance constraints are of the form $\langle TERM \ REL \ TERM \rangle$ where REL is one of $<$, $<=$, $=$, $/=$, $>=$, or $>$, and $TERM$ is one of

- $EVENT.LOCATION$ – location of the triggering event

- EVENT.SEVERITY – severity of the triggering event (an integer 0-10)
- EVENT.NUMBER – number of people affected by the triggering event
- EVENT.TYPE – type of the triggering event
- EPIDEMIC – constant representing the epidemic event type
- MUDSLIDE – constant representing the mudslide event type
- EVACUATION – constant representing the evacuation event type
- TASK.PRIORITY – priority of the currently executing task (an integer 0-10)
- TASK.LOCATION – location of the currently executing task
- TRANSPORT-TYPE – transport-type role of a potential plan
- HELICOPTER – constant representing the helicopter transport type
- TRUCK – constant representing the truck transport type
- CITY- N – constant representing the city/town with id N in the simulator
- N – integer N

4.3 Interacting with TIGER – An Example Script

The script for a demonstration of the TIGER system shows the use of adjustable autonomy and guidance rules in guiding agent behavior. In this scenario, TIGER has at its disposal five trucks and one helicopter agent. TIGER is given an assessment task to execute, consisting of surveying a number of towns. While pursuing this goal, TIGER receives notification of an emergency. Based on the guidance rules that the J2 commander has set up, a nearby agent chooses to respond to the emergency by dropping an existing survey task. Based on the level of autonomy set, the agent will ask the J2 commander for permission before doing so. The agent has choices in how to respond to the emergency, and these are also governed by guidance rules and autonomy constraints.

Action: In a freshly started system, allocate the J2 role to TIGER by clicking the button “Allocate J2 role” on the “Demo Events” window.

Having been allocated the J2 role, the J2 commander would normally set up the appropriate level of autonomy, guidance, and reporting by using the permission, guidance, and reporting windows. The default settings for each of these windows are suitable for this example and need not be changed (but can be if you want to vary the demonstration).

Action: Review the autonomy, guidance, and reporting settings using the windows access through the “Set Permissions”, “Set Guidance”, and “Customize Reporting” buttons on the Main menu.

We now want to send the J2 a request to perform an assessment on a nearby region (set of towns).

Action: Click the “Provide assessment task” button on the “Demo Events” window.

The agents corresponding to the five trucks and one helicopter that are under the command of the J2 start taking on tasks to support this assessment request.

Action: After all the vehicles have started their tasks, send a mudslide message from the “Demo Events” window by clicking on the “Mudslide event” button.

In the “Notifications” pane of the “Reports Display” window, notes will appear indicating that guidance rules have become applicable. First the rule “Use HELICOPTER when responding to emergency” is used to prefer a helicopter response over a truck response. Then the rule “Respond to emergencies involving more than 10 people with severity greater than the current task priority” triggers, causing the agent to prefer to adopt the emergency task (rather than ignore it), in the process aborting the existing survey task.

The action of dropping a survey task is something about which the agent must consult the commander, according to the permission settings. This causes a permission window to pop up asking whether to allow or reject the action.

Action: Click on the “Allow” button to accept the action.

The response to a mudslide may involve either treating the injured at the site or evacuating them to another site. In the “Notifications” pane, we see that the rule “Use external response when responding to medical emergencies involving more than 20 people” is triggered and causes the helicopter agent to favor evacuation.

When the helicopter reaches the site of the emergency, there is a choice of medevac sites to use (based on the location of the emergency). Now the consultation rule “Consult supervisor when selecting a medevac site” is triggered and a “Consultation Request” window appears, asking for guidance on which of the possible medevac sites to use.

Action: Click on any of the numbered choice buttons.

During all this activity, one or more of the trucks may have discovered that the road conditions are such that the towns they are trying to reach are inaccessible. If this occurs, the agent will attempt to abandon the task. Given the default autonomy settings, the agent will be required to request permission to abandon the task (as opposed to continue trying in the hope that the road will clear). This results in “Request for Permission” windows appearing.

Action: Click “Allow” to allow the agents to abandon these tasks. Should the number of such requests start to overwhelm the J2 commander, it is possible to reset the autonomy levels in the “Permissions” window so as to allow the agents to abandon tasks at their own discretion.

4.4 Quitting

To avoid leaving Tcl/Tk processes still executing after quitting TIGER, you must quit the TIGER interface before exiting TIGER.

1. Select “Quit” from the “Main Menu” TIGER interface window to kill the user interface.
2. Click on the close box of the TIGER Lisp interface to kill the TIGER process.
3. Exit from SimSpy2 if it is running.
4. Select the simulator window and type control-C to kill the simulator process.

If the Tcl/Tk processes are still running after Lisp has exited, the processes can be terminated by using the Windows Task Manager.

4.5 TIGER Initialization

TIGER loads the initial “beliefs” of the agents in TIGER from the file `tasking-db.lisp` in the directory `TIGER/miata/1.2/lisp/`. Initial beliefs that are easily customizable are

- Possible medevac sites for a given emergency location, which are specified as facts of the form `(MEDEVAC-SITE "Yorito" "El Porvenir")` (stating that “El Porvenir” is a valid medevac site for an emergency in “Yorito”).
- The initial allocations of vehicles (i.e., trucks) and helicopters, which are specified as facts of the form `(INITIAL-RESOURCES (VEHICLE "ENRIQUE SOTO CANO" 5))` (stating that TIGER starts off with five trucks located at “ENRIQUE SOTO CANO”).

4.6 Plan Execution tracing

The plans for achieving survey and emergency tasks are specified using the PRS plan language [10]. To better understand the execution of the plans, consult the PRS manual [7] on how to enable tracing the execution of plans, changes to an agent’s beliefs, and messages that an agent receives.

Acknowledgments

The author thanks Eric Hsu for his contributions in developing the TIGER interface, and Sebastian Thrun and his group at CMU for providing the MAPLE simulator. This work was supported by DARPA under the supervision of Air Force Research Laboratory contract F30602-98-C-0160.

References

- [1] Pete Bonasso. Issues in providing adjustable autonomy in the 3T architecture. In *Proceedings of the AAAI Spring Symposium on Agents with Adjustable Autonomy*, 1999.
- [2] H. Chalupsky, Y. Gil, C. A. Knoblock, K. Lerman, J. Oh, D.V. 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.
- [3] George Ferguson and James Allen. TRIPS: Towards a mixed-initiative planning assistant. In *Proceedings of the AIPS Workshop on Interactive and Collaborative Planning*, 1998.
- [4] Michael P. Georgeff and François Félix Ingrand. Decision-making in an embedded reasoning system. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.
- [5] 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.
- [6] K. L. Myers and D. N. Morley. Policy-based agent directability. In Henry Hexmoor, Rino Falcone, and Cristiano Castelfranchi, editors, *Agent Autonomy*. Kluwer Academic Publishers, 2002. To Appear.
- [7] Karen L. Myers. *User's Guide for the Procedural Reasoning System*. Artificial Intelligence Center, SRI International, Menlo Park, CA, 1993.
- [8] Karen L. Myers and David N. Morley. Human directability of agents. In *Proceedings of the First International Conference on Knowledge Capture*, 2001.
- [9] Debra Schreckenghost, Jane Malin, Carroll Thronesbery, Grayum Watts, and Land 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.
- [10] David E. Wilkins and Karen L. Myers. A common knowledge representation for plan generation and reactive execution. *Journal of Logic and Computation*, 5(6), 1995.