GTrans: An Application for Managing Mixed-Initiative Collaborative Planning during Emergency Response Situations

Trivikram Immaneni and Michael T. Cox
Wright State University
Department of Computer Science & Engineering
Dayton, OH 45435-0001
{timmanen, mcox}@cs.wright.edu

Abstract

This paper describes the design of a mixed-initiative collaborative planning system called GTrans. GTrans is a distributed application in which multiple, remote agents collaborate to jointly solve a problem. The system allows the users to interact with semi-autonomous planning agents and with each other. When solving a given problem, resource constraints often prevent perfect plans from being assembled that achieve all goals. In such cases, users are able to shift resources and to shift the goals themselves so that equilibrium can be achieved to maximize the solutions. To assist collaboration, GTrans controls what each user views through a shared object mechanism. Currently the mechanism supports static selective filtering, but it provides an extensible framework that will enable dynamic filtering.

Keywords: Design of collaborative systems, intelligent agents in collaborative applications, shared objects, mixed-initiative planning, goal transformations.

1 INTRODUCTION

GTrans (Goal Transformations) [1-3] is a mixed-initiative planning system that has been designed to directly support the concept of goal transformation [4] or change. It presents planning tasks to the user as a goal manipulation problem rather than a problem of search [2, 5, 6]. GTrans hides the underlying planner and representations from the user and provides the user with simple mechanisms with the help of which she can actively participate in the planning process. However most of our research has focused on single user / single planner interaction. Here we present preliminary work toward the support of collaborative mixed-initiative planning between multiple users.

The most straight forward approach to supporting multiple human agents is to present a single unified view of a planning scenario, associated resources and shared goals. With realistic planning situations, however, many of the resources and details present will not be relevant to most users of the system. We have implemented a shared-object mechanism that begins to control the individual views each user maintains. The mechanism allows the establishment of ownership of resources and goals so that users have separate viewing and control properties for each object in a planning environment. The mechanism also supports the manual transfer of object ownership for shifting the assignment of resources. Using this mechanism selective filtering is established so that users are not overwhelmed by inappropriate information.

While adding such a capability, it is also necessary to maintain the overall metaphor of planning as a goal manipulation process. In this paper we present a multi-user version of GTrans that uses a newly implemented emergency response domain. In this domain planners can explore tradeoffs between partial goal achievement (goal shift) and resource reassignment (ownership shift). We present a simple scenario to illustrate these ideas and to highlight the implementation. Moreover, we describe how the current static filtering of object views can be made more flexible by the addition of dynamic object exposure that is triggered by inappropriate planning decisions made by collaborative teammates.

Section 2 describes the basic architecture of GTrans and the major components of the system. Section 3 describes the various modes in which GTrans operates. Section 4 describes the concept of shared objects and Section 5 describes some of the research issues that we are planning to explore in the future. We conclude with Section 6.

2 THE GTRANS SYSTEM

The architecture of the GTrans system[1] is shown in Figure 1. The GTrans Server, GTrans User Agent and PRODIGY/AGENT are the major components in the system. The basic planning process for a user is to create a problem using the graphical user interface of the GTrans User Agent and to send it to the underlying Prodigy/Agent planner. If all is well, the planner will generate a successful

[1]The GTrans home page is located at the following URL. www.cs.wright.edu/~mcox/GTrans

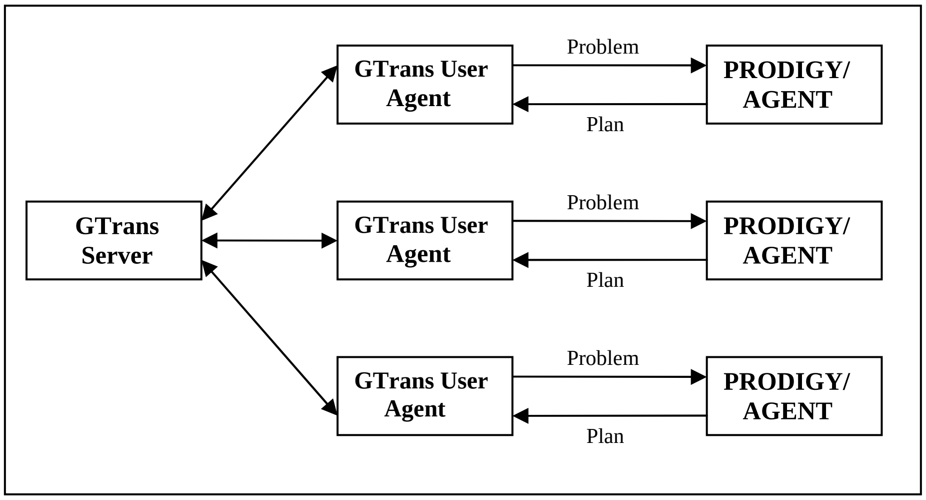


Figure 1. The GTrans Architecture

plan for the problem and return it to the User Agent, which will then display it to the user. If the planner cannot create a plan (usually due to the lack of adequate resources) it will return the message "No Plan" to the User Agent. When this happens, the user must either change the goals or acquire the necessary resources from other GTrans User Agents. The User can then send the (modified) problem back to PRODIGY for planning. The GTrans Server acts like a central mediator that helps multiple, remote GTrans Agents to coordinate with each other and share each other's resources. The following is a brief description of each of the major components of GTrans.

2.1 Prodigy/Agent

Prodigy/Agent [7, 8][2] (written in Allegro Common Lisp and compatible with versions 5.0.1 or higher) consists of the PRODIGY planner and a wrapper program. The wrapper program acts as a software interface between the PRODIGY planner and the GTrans User Agent. It allows the GTrans User Agent to communicate with the PRODIGY planner through a protocol implemented in the Knowledge Query and Manipulation Language or KQML [9]. The two modules exchange KQML performatives using socket communication.

PRODIGY [10, 11] is a domain-independent, nonlinear state-space planner implemented originally at Carnegie Mellon University. It searches for a sequence of actions that transform the environment from an initial-state into a final-state containing the goal state. Like all state-space planners, its problem specification includes a set of objects existing in the planning environment, the initial state of the environment, and the goal state that needs to be achieved by the plan. As of now only four of the many PRODIGY domains have been implemented to work with GTrans - the "military," "blocksworld," "package delivery" and "emergency response."

[2] The Prodigy/Agent home page is located at the following URL www.cs.wright.edu/~mcox/Prodigy-Agent

2.2 The GTrans User Agent

This module has been implemented using Java Version 1.2. The user interacts with the system using the graphical user interface (GUI) of the GTrans User Agent. This GUI consists of a main menu bar with dynamic menus and a work area (called the canvas) in which the user builds the problem. Building the problem and sending it to the planner is a simple, interactive, menu-driven process. For example, the user can create objects by choosing an object type from the "Object" menu of the GUI and clicking on the canvas. Objects can be moved by dragging and dropping. The GUI provides the user with a simple menu-driven mechanism to facilitate goal change.

GTrans is a multi-agent system in which several planning agents can jointly solve a problem. The GTrans Agent can operate in the three distinct operational modes of "Separate Planning" (default mode), "Info-sharing" and "Joint Planning" modes. The user can choose from a drop-down menu, the planning mode in which she wants to operate. If the user chooses to work in the Separate Planning mode, then she will perform separate, stand-alone planning without any access to other agents' information. In the Info-Sharing mode, the user will be able to view the other agents' information, but she will not be able to use the other agents' resources to solve her problem. In the Joint Planning mode, the user will not only be able to view other agents' information but will also be able to use their resources to solve a problem.

In the Info-Sharing and Joint modes, the GUI will act like a typical whiteboard application. Each event that occurs with an agent (e.g., object creation, object deletion) is first sent to the GTrans Server. The server then broadcasts it to all the other agents. The agents communicate with the central GTrans server (and vice-versa) using Java RMI. If the agent receiving this broadcast information is in the Separate Planning mode, it simply discards the information. But if the agent is in the Info-Sharing or Joint Planning modes, it receives the information and updates its data structures to reflect the new event. The objects belonging to other agents are labeled red in order to help the user distinguish them from local objects. The process of creating a problem, changing goals, and the operation of GTrans in various modes is discussed in the next section with the help of an example scenario.

2.3 The GTrans Server

This module has been implemented using Java Version 1.2. The GTrans Server acts like a central coordinator for the agents in the system. It communicates with the agents using the Java remote method invocation mechanism. The server is also responsible for assigning a unique identification number to every agent in the system. When an agent is started, it first contacts the server and requests its identification. Most of the events occurring at an agent are sent to the server, which then broadcasts the information to all the other agents.

The GTrans Server also acts as a repository of shared resources. Any agent can request the use of these resources. The server is responsible for coordinating and controlling the use of these shared resources by the agents. The concept of shared resources and the role of the server in controlling them are discussed with an example in Section 4.

3 THE OPERATION OF GTRANS

As part of an ongoing project with Ball Aerospace and Science Applications International Corporation (SAIC), we are currently integrating the GTrans system with Ball's Knowledge Kinetics© software and with SAIC's Consequences Assessment Tool Set (CATS). Also as a part of this project, we are implementing a simple emergency management planning domain in GTrans. This section describes the working of the GTrans system using an example scenario from this domain.

The objects in a given domain are broadly divided into the two categories of stationary and mobile. The mobile object types in the emergency response domain include squad car, ambulance, fire truck, victim, sheriff car, SWAT team and a Hazardous Materials (HAZ-MAT) team. The stationary objects include city, scene-of-incident, police station, fire station, hospital, HAZMAT station, chemical-spill, fire, and sheriff station. A typical scenario in this domain is shown in figure 2.

This hypothetical scenario is a result of a tornado. We can see two incidents, two fires, two fire trucks, two squad cars, a HAZMAT team, an ambulance, a fire station, a hospital, a HAZMAT station and the city called Xenia. As mentioned before, GTrans provides the user with simple GUI driven mechanism to create objects. Once an object has

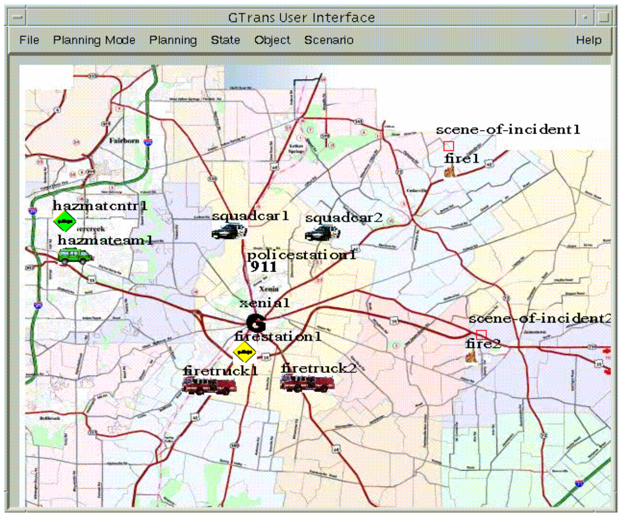


Figure 2. A scenario in the emergency response domain

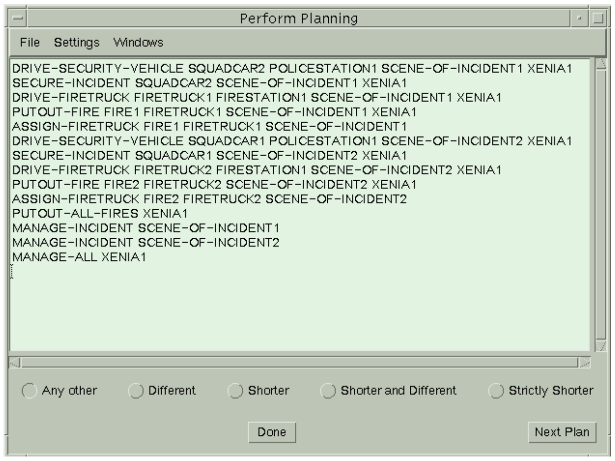
been created, GTrans dynamically extracts all the possible initial states that the object can be in from the domain information. Likewise, GTrans derives all the possible goals that can be associated with the object from the domain information. The user can then set the initial state and the goal state of the object by using simple GUI mechanisms. For example, in the above scenario, the following goals were established on xenia1.

(outcome-putout-fires xenia1)
(is-managed-all xenia1)

The domain has been organized so that to put out a fire at a scene-of-incident, the scene must first be secured by a security vehicle (i.e., squad car or sheriff car). Once the scene is secured, a fire truck must be driven to the scene and put out the fire. The domain has been designed such that a single fire truck can guarantee to put out one fire and contain an arbitrary number of others. To fulfill the goal (outcome-putout-fires xenia1), all the fires in xenia1 must be extinguished. The goal (is-managed-all xenia1) is fulfilled when all the incidents in xenia have been managed.

Enough resources exist in figure 2 to achieve these goals. Two squad cars can secure the two scenes and two fire trucks can put out the two fires. After building the scenario, the user can save the scenario for later use. She can then send the problem to the planner using appropriate menus. The underlying planner will try to generate a plan and will return any results. If the plan is not acceptable to the user, Prodigy/Agent can generate an additional plan at the user's request. The user can also request different and shorter plans. Figure 3 shows the plan generated by the planner for the scenario shown in figure 2. The plan is to use squad-car2 and squad-car1 to secure scene-of-incident1 and scene-of-incident2 respectively and to use firetruck1 and firetruck2 to put out the fires fire1 and fire2 respectively.

The GUI also provides the user with an option to associate specific resources with goals. When a mobile object is dragged near a stationary object, GTrans extracts



```
DRIVE-SECURITY-VEHICLE SQUADCAR2 POLICESTATION1 SCENE-OF-INCIDENT1 XENIA1
SECURE-INCIDENT SQUADCAR2 SCENE-OF-INCIDENT1 XENIA1
DRIVE-FIRETRUCK FIRETRUCK1 FIRESTATION1 SCENE-OF-INCIDENT1 XENIA1
PUTOUT-FIRE FIRE1 FIRETRUCK1 SCENE-OF-INCIDENT1 XENIA1
ASSIGN-FIRETRUCK FIRE1 FIRETRUCK1 SCENE-OF-INCIDENT1
DRIVE-SECURITY-VEHICLE SQUADCAR1 POLICESTATION1 SCENE-OF-INCIDENT2 XENIA1
SECURE-INCIDENT SQUADCAR1 SCENE-OF-INCIDENT2 XENIA1
DRIVE-FIRETRUCK FIRETRUCK2 FIRESTATION1 SCENE-OF-INCIDENT2 XENIA1
PUTOUT-FIRE FIRE2 FIRETRUCK2 SCENE-OF-INCIDENT2 XENIA1
ASSIGN-FIRETRUCK FIRE2 FIRETRUCK2 SCENE-OF-INCIDENT2
PUTOUT-ALL-FIRES XENIA1
MANAGE-INCIDENT SCENE-OF-INCIDENT1
MANAGE-INCIDENT SCENE-OF-INCIDENT2
MANAGE-ALL XENIA1
```

Figure 3. Plan generated by PRODIGY for fig.2 scenario

all the possible goal states that "connect" the two objects from the domain information. GTrans then presents this list to the user from which she can select the appropriate goal. In the above example, if the user wants to make sure that squad-car1 is used to secure scene-of-incident1, then she can drag the squad-car1 to the scene-of-incident1 and choose the state "at-squadcar1 scene-of-incident1" from the menu that appears. This will set another goal state that stipulates that squadcar1 has to be at scene-of-incident1.

In the above scenario, the planner had sufficient resources to solve the problem and hence it was able to generate a plan. Now consider a scenario in which the planner does *not* have sufficient resources to generate a plan. The scenario is the same as above but with one additional incident, scene-of-incident3. We have a chemical-spill at scene-of-incident3. The domain has been designed such that, to manage a chemical-spill, security personnel must be present at the scene (i.e., at least one squad car or a fire truck must be present). Also the HAZMAT team must arrive at the scene to manage it. In this scenario, we do not have sufficient resources to solve the problem. We need another security-vehicle or fire truck. If this problem is sent to Prodigy/Agent with the same goals as before, the planner will return the message "No Plan." At this point, the user can either transform the goals or acquire additional resources to solve the problem. For example, the user can lower her expectations by changing the goal of putting out all the fires to the goal of containing the fires so that one fire truck can handle the new incident. Alternatively, the user may decide to let the fires burn. Figure 4 depicts the pop-up menu with which the user changes the goal.

Once the goal has been transformed, the user can resend the problem to the underlying planner. In this case, the planner generates a successful plan. The plan is to use fire-truck1 to put out fire1 and contain fire2 and to use fire-truck2 to take care of the spill. The HAZMAT Team is to be driven to the spill to stabilize and dispose the spill. As an alternative to goal change, the user can change to the joint -

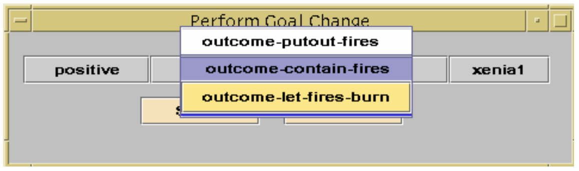


Figure 4. Performing Goal Change

mode and collaborate with other users to solve this problem. In the Joint mode, the user will see the objects of the other agents in the system. The user can also see the state information and the goal information of the other agents in the system. In this mode, the user can use the objects belonging to the other agents to solve her problem. While in the joint-mode, every operation performed by the remote users in the system (adding an object, setting a state, setting a goal etc.,) can be immediately seen by the user.

For the above example, let us consider another agent in the system, working with a highly simplified scenario. That is, the agent has a sheriff car stationed at a sheriff station, but the agent does not have any incidents to manage. Now the user need not make any goal change, because in the joint mode she can use the extra sheriff car to solve her problem.
This is a highly simplified example to demonstrate the operation of GTrans. In practice, it may so happen that the second agent might have a fire and no fire truck. In such a case, the two agents must collaborate. The first agent lowers its expectations so that the second agent can use the first agent's fire truck to extinguish the fire.

To illustrate this concept better, consider the situation where the first agent has only one squad car (instead of the two shown in fig. 2). Let us assume that a chemical-spill has occurred in the second Agent's scenario. Figure 5 shows second Agent's GUI.

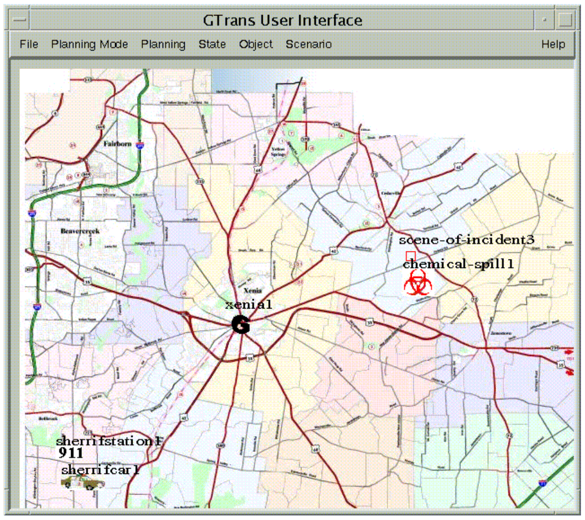


Figure 5. Second Agent's scenario

That is, the first Agent has two fires but one only security vehicle whereas the second Agent has a security vehicle and a chemical-spill. The first Agent needs the sheriff car from the second Agent and the second Agent needs the HAZMAT Team from the first Agent. In this case the two agents must collaborate and negotiate on the goals to decide how their resources can be used to solve the problems. For example, the first user may lower her expectations and change her goal of "outcome-putout-fires" to "outcome-contain-fires" so that one of the fire trucks can be used by the second Agent to take care of the spill. In return the Agent can use the second Agent's sheriff car to secure its scene-of-incident.

4 SHARED OBJECTS

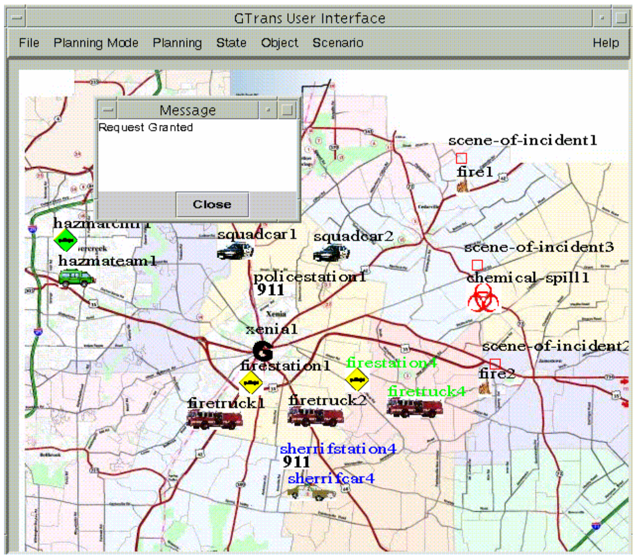
One current research issue is the idea of filtering the information presented to the user. In the joint-mode, the user can presently view all objects belonging to all of the other agents in the system. Ideally, the system should display only those remote objects that are relevant to the problem the user is trying to solve.

Another issue we are currently investigating is the idea of ownership of objects. At present, while planning in the joint-mode, the user assumes that the objects belonging to other agents are readily available for her use. In the real, world this might be an unrealistic assumption. For example in order to use a sheriff car, the city must first get permission from state authorities. The state authority may or may not give the permission in which case the user must consider an alternate plan.

To address these issues, we are incorporating the idea of shared objects in GTrans. We have designed the server to act as a repository of objects that can be used by any of the agents in the GTrans system. The shared objects are labeled green on the canvas to help distinguish them from other objects. Not every user can view all shared objects. Each shared object is associated with a list of agents, and only these agents receive information about the object. The server reads the shared object information from a text file and stores the objects in its data structures. The "visibility" information is also read from the same text file. The users can request only those shared objects that are visible on their GUI. In fact they do not have any knowledge of the other shared objects. In order to use a shared object, the user must first request the server for permission to use the object.

The user can request a shared object using a menu-driven mechanism. When the server receives this request, it first checks to see if any other agent is using the object. If no other agent is using the object, the server grants the request and sends the message "Permission Granted" to the agent. At this point, the label of the shared object being requested turns blue (only on the requesting agent's GUI), indicating that the agent can now use the object in its plans.

Figure 6 shows the original scenario along with a sheriff car and the (third) fire truck. The sheriff car has been requested by the user and the server has granted the user permission to use the sheriff car. At this point, if any other agent requests the same object, the server denies permission with the message "Object in use…Permission denied." Once



Request Granted

Figure 6. Message from the server granting the request for the sheriff car

the agent has finished using the object, it must release the object so that others can use it. The data structures in the server are protected from simultaneous accesses to ensure consistency. In GTrans, the "resources" are usually mobile objects. Each mobile object is associated with a stationary object. For example, the sheriff car is attached to a sheriff station. For this reason, each shared object is actually a pair of GTrans objects. This concept of shared objects ensures that a resource is used by only one agent at a time and it also helps the agents to take resource availability into consideration while planning. The shared objects (that are available to the agent) stay hidden until the user wishes to see them. This ensures that they do not hinder the process of planning with local resources.

5 FUTURE RESEARCH

This paper discusses a mechanism whereby a static object view and ownership determines what individual planners can see and control in the collaborative planning process. Although the shared objects can change through manual resource requests, the view individuals possess is not sensitive to changes in the environment due to exogenous events or to planning steps chosen by collaborative teammates. For example one teammate may decide to establish a roadblock to secure the site of an incident, but by doing so this action may prevent the passage of an ambulance necessary for the achievement of another teammate's goal to assist burn victims. Ideally the system should be able to detect this goal interaction, to dynamically expose the squad-car involved in the roadblock and the vehicle involved in transit, and thereby to initiate negotiation between the two collaborators.

Previous research on artificial multiagent systems using Prodigy/Agent [12] does detect such goal interaction. Rationale-based planning monitors [13] provide such coordination by watching for changes in the state of those conditions responsible for action selection and by broadcasting each planning operator‘s effects when the operator is applied during the search process of PRODIGY. For example, if one agent decides to use a Drive-Ambulance operator whose precondition is that the road upon which the transit occurs is open, Prodigy/Agent establishes a monitor to watch that the state does not change. Then if another agent decides to block a road with a squad car, when the operator is applied to the current state, the state change is broadcast to all monitors. The initial agent can therefore trap the effect of the other agent if the two roads are the same and initiate a response. As a result the next issue we intend to investigate is the use of such planning monitors in the service of dynamic filtering and exposure of objects in the individual views presented by GTrans.

6 CONCLUSION

Although this research and the prototype developed under it is only preliminary, it has provided a number of insights into the problems and approaches associated with mixed-initiative collaboration. Using a fully automated planner in actual emergency situations is impractical. The writer of a domain cannot possibly imagine all the possible parameters that might need to be taken into consideration to solve a problem in an emergency situation. Mixed-initiative planning is ideal for these types of situations. The human user plays a very active role in the planning process and can serve to compensate for the shortcomings of the planner itself. The system serves as a tool for the user to better assess the situation thereby helping her make critical decisions. A collaborative system like GTrans not only helps multiple remote users to get a better perspective on the situation but also lays a foundation upon which the decision makers can negotiate. They thus can compromise on their individual goals to solve the overall goal of minimizing the damage caused by an emergency such as a tornado.

ACKNOWLEDGEMENTS

This research has been funded by a grant from the Information Technology Research Institute (ITRI) and a grant from SAIC. The authors wish to thank N. N. Schneider of Ball Aerospace and Richard Henderson of SAIC for their valuable help in implementing an integration of GTrans into more realistic set of scenarios than would otherwise be possible using GTrans alone. We also thank Vance Saunders (Ball), Denny Kirlin (SAIC) and Brian Beebe (SAIC) for comments and feedback on the project.

REFERENCES

[1] Cox, M., B.Kerekez, C.Srinivas, G.Edwin, and W.Archer. 2000. “Toward agent-based mixed initiative interfaces.” In *Proceedings of the 2000 International Conference on Artificial Intelligence*, ed. H. R. Arabnia, 1:309-315. CSREA Press.

[2] Zhang, C. 2002. “Cognitive Models for Mixed- Initiative Planning.” Master’s thesis, Wright State University.

[3] Zhang, C., M. T. Cox, and T. Immaneni. 2002. “GTrans version 2.1 User Manual and Reference.” Technical Report WSU-CS-02-02. Department of Computer Science and Engineering, Wright State University.

[4] Cox, M. T., and M. M. Veloso. 1998. ”Goal transformations in continuous planning.” In *Proceedings of the 1998 AAAI Fall Symposium on Distributed Continual Planning*, ed. M. desJardins, 23-30. Menlo Park, CA: AAAI Press / The MIT Press.

[5] Cox, M. T. 2000. “A conflict of metaphors: Modeling the planning process.” In *Proceedings of 2000 Summer Computer Simulation Conference*, 666-671. San Diego: The Society for Computer Simulation International.

[6] Cox, M. T. 2003. ”Planning as mixed-initiative goal manipulation.” In *Proceedings of the Workshop on Mixed-Initiative Intelligent Systems at the 18th International Conference on Artificial Intelligence.* Menlo Park, CA: AAAI Press.

[7] Cox, M. T., G. Edwin, K. Balasubramaniam, and M. Elahi. 2001.”Multiagent goal transformation and mixed initiative planning using Prodigy/Agent.” In *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics*, ed. N. Callaos, B. Sanchez, L. H. Encinas, and J. G. Busse, 7:1-6. Orlando, FL: International Institute of Informatics and Systemics.

[8] Elahi, M. M. 2003. “A distributed planning approach using multiagent goal transformations.” Master’s thesis, Wright State University.

[9] Finin, Tim, Don McKay, and Rich Fritzson. 1992. “An Overview of KQML: A Knowledge Query and Manipulation Language.” Technical Report, Department of Computer Science, University of Maryland.

[10] Carbonell, J. G., J. Blythe, O. Etzioni, Y. Gil, R. Joseph, D. Kahn, C. Knoblock, S. Minton, A. Perez, S. Reilly, M. M. Veloso, and X. Wang. 1992. “PRODIGY4.0: The Manual and Tutorial.” Technical Report CMU-CS-92-150. Computer Science Department, Carnegie Mellon University.

[11] Veloso, M. M., J. G., Carbonell, A. Perez, D. Borrajo, E. Fink, and J. Blythe. 1995. “Integrating planning and learning: The PRODIGY architecture.” *Journal of Theoretical and Experimental Artificial Intelligenc*e 7(1): 81-120.

[12]Edwin, G., and M. T. Cox. 2001. “Resource coordination in single agent and multiagent systems.” In *Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence*, 18-24. Los Alamitos, CA: IEEE Computer Society.

[13] Veloso, M. M., M. E. Pollack, and M. T. Cox. 1998. “Rationale-based monitoring for continuous planning in dynamic environments.” In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, ed. R. Simmons, M. Veloso, and S. Smith, 171-179. Menlo Park, CA: AAAI Press.