Integrating Agent-Based Mixed-Initiative Control With An Existing Multi-Agent Planning System

Mark Burstein
BBN Technologies
burstein@bbn.com

George Ferguson and James Allen
University of Rochester
{ferguson,james}@cs.rochester.edu

The University of Rochester
Computer Science Department
Rochester, New York 14627

Technical Report 729

January 2003

This work is supported by The Defense Advanced Research Projects Agency (DARPA) under contracts F30602-98-C-0168 and F30602-98-2-0133.

Abstract

One of the less appreciated obstacles to scaling multi-agent systems is understanding the impact of the role(s) that people will play in those systems. As we try to adapt existing software tools and agent-based applications to play supportive roles in larger multi-agent systems, we must develop strategies for coordinating not only the problem solving behavior of these agent communities, but also their information sharing and interactive behavior. Our research interest is in mixed-initiative control of intelligent systems [Burstein and McDermott, 1996; Burstein *et al.*, 1998; Ferguson *et al.*, 1996a] and, in particular, of interactive planning systems comprised of a heterogeneous collection of software agents. In this paper, we describe our experience constructing a prototype tool combining elements of TRIPS [Ferguson and Allen, 1998], an interactive, mixed-initiative agent-based planning architecture using spoken natural language dialogue, with the CAMPS Mission Planner, an interactive airlift scheduling tool developed for the Air Force [Emerson and Burstein, 1999], together with some related resource management agents representing other parts of the airlift planning organization. The latter scheduling tools were not originally designed to participate as part of a mixed-initiative, interactive agent community, but rather were designed for direct user interaction through their own GUIs. We describe some requirements revealed by this effort for effective mixed-initiative interaction in such an environment, including the role of explanation, the need for contextual information sharing among the agents, and our approach to intelligent invocation and integration of available agent capabilities.

1 Introduction

One of the less appreciated obstacles to scaling multi-agent systems is understanding how people will play roles in those systems, and the nature of their interactions with them. We expect that many multi-agent systems will evolve out of existing software tools, suitably "agentized" so that they have greater access to other software agent resources, and to other classes of users. To develop these kinds of agent-based applications, playing supportive roles in larger multi-agent systems, we must develop strategies for coordinating not only the problem solving behavior of these agent communities, but also their information sharing and interactive behavior with users.

Our research is on mixed-initiative (user/agent) control of intelligent systems [Burstein and McDermott, 1996; Burstein *et al.*, 1998; Ferguson *et al.*, 1996a], and, in particular, of interactive planning systems comprised of a heterogeneous collection of software agents. Over the past year, we have been experimenting in the space of mixed-initiative, multi-agent planning systems. We have developed a prototype mixed-initiative planning tool for airlift scheduling by integrating elements of TRIPS, The Rochester Interactive Planning System, an agent-based, interactive, mixed-initiative planning system using spoken natural language dialogue [Ferguson and Allen, 1998], with the CAMPS Mission Planner, an interactive airlift scheduling tool developed for the Air Force [Emerson and Burstein, 1999], together with some related resource management agents representing other parts of the airlift planning organization.

This effort had a number of goals. First, we wished to demonstrate the relative ease with which the TRIPS agent architecture could be adapted to a new planning domain, and to interaction with a new back-end planner. Second, we sought to understand what would be required for an effective agent ACL interface to a scheduling or planning tool that had its own GUI and was not developed for multi-modal mixed-initiative interaction. Third, we wished to develop a model for the problem solving agent that could mediate between the user and a set of "back office" planning agents. The last of these goals is an ongoing activity.

2 Background

Our work involves the integration of three existing systems, each of which is described briefly in this section.

TRIPS, The Rochester Interactive Planner System [Ferguson and Allen, 1998] is a multi-agent system that includes agents for interacting with the user in spoken natural language (*e.g.*, speech recognition, parsing, reference resolution, discourse management, speech generation, *etc.*) as well as knowledge-based agents

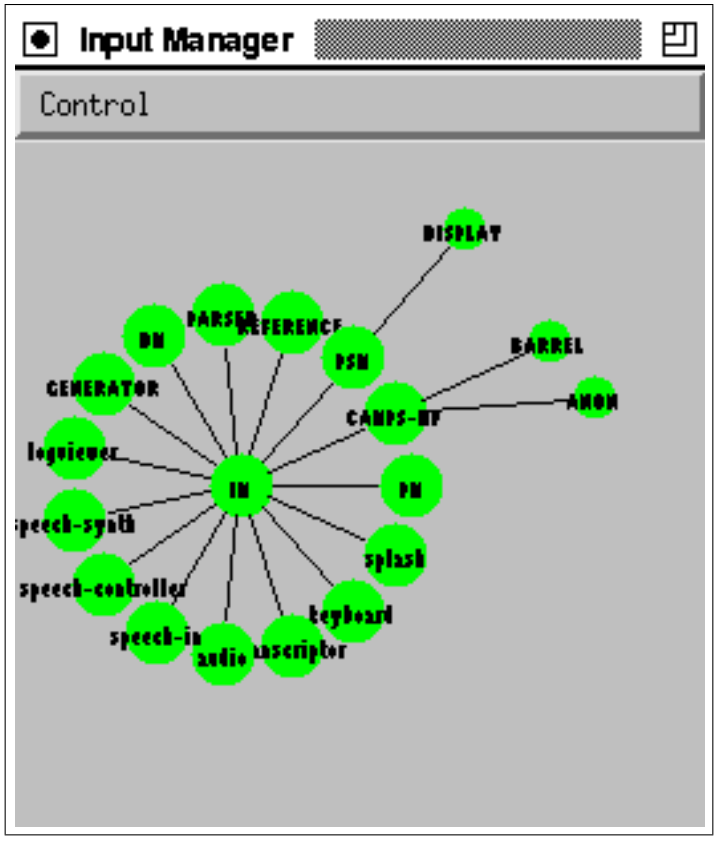


Figure 1: TRIPS Facilitator display

that provide task and domain expertise to assist the user (*e.g.*, planning, plan recognition, route-finding, scheduling, *etc.*). It is the latest in a series of mixed-initiative systems produced by an extended multi-year research program at the University of Rochester that began with the TRAINS system [Allen *et al.*, 1995; Ferguson *et al.*, 1996b]. The agents collectively provide a multi-modal interface by which users can discuss and develop plans through mixed-initiative interaction with what appears to them to be a single intelligent agent. The agents interact to produce this behaviour by exchanging messages using the KQML agent communication language [Finin *et al.*, 1993; Labrou and Finin, 1997], operating in a hub-based architecture provided by the TRIPS Facilitator. Figure 1 shows the Facilitator's GUI view as it appears in the integrated TRIPS/CAMPS system discussed in this paper (secondary nodes indicate a proxying relationship between agents). Figure 2 shows the logical organization of the agents involved in the TRIPS/CAMPS system.

Previously, TRIPS/TRAINS has been demonstrated for planning in domains such as non-combatant evacuations (NEOs), where a user and the system would

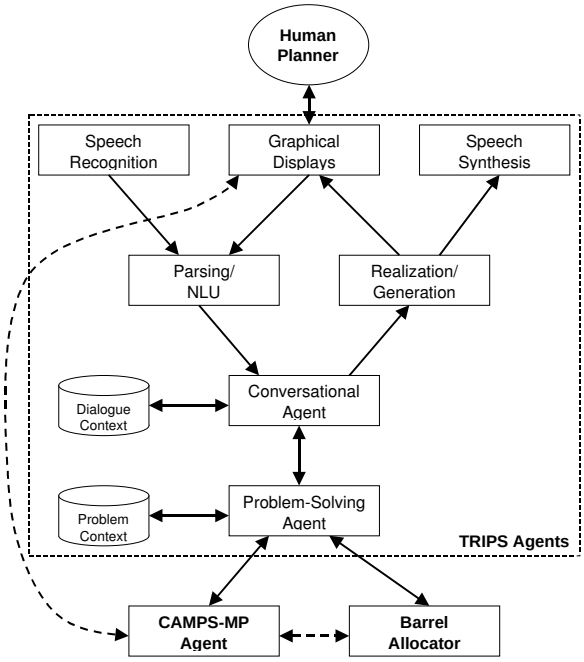


Figure 2: TRIPS/CAMPS architecture (logical organization)

jointly develop a plan to evacuate people from a fictitious island (Pacifica) in the face of an approaching hurricane. The user's task was to plan the evacuation using a variety of vehicles with varying capabilities, under constraints such as time, cost, weather, *etc.* The modality management agents (speech and NL processing, gesture capture) translate user actions into a set of possible *communicative acts* that are interpreted by the *Conversational Agent*. This agent interacts with the *Problem-Solving Agent* to find an plausible interpretation of what the user has just said and/or done. They then jointly determine an appropriate response by coordinating domain-level reasoners such as planners and schedulers. The Problem Solving Agent manages the invocation of the back-end agents and coordinates their responses.

TRIPS supports a wide range of speech acts, ranging from direct requests (*e.g.*, "show me the map"), questions ("Where are the transports?"), suggestions ("Let's use a helicopter instead."), acceptances, rejections, and a range of social acts. The Conversational Agent is domain independent. It is driven by a set of rules that identify possible interpretations intended by a user, and plans an appropriate kind of system response for each. Each interpretation/response is ranked and a single response is selected. The Problem Solving agent (PS) is responsible for maintaining the problem solving context, interacting with the Conversational Agent to decide which interpretations of user actions are plausible, and plan for the execution of the selected action by communicating with the appropriate back end agent(s).

The specialized back-end agents in TRIPS-98 and earlier incarnations included

U: Show me a map of pacifica.
T: Ok. *(shows the map)*
U: Where are the people?
T: There are two groups of people at Exodus, two at Calypso, ...
U: Use a truck to get the people from Calypso to Delta.
T: Ok. *(show timeline view of plan under development in Plan Viewer)*
U: How long will that take?
T: It will take four hours and fifty minutes.
U: What if we went along the coast instead?
T: That option will take six hours and thirty-seven minutes under normal conditions.
U: Forget it.
...

Figure 3: Sample Dialogue with TRIPS-98 (excerpt)

an incremental (repair-oriented) temporal planner, a route finder, a scheduler, a simulator that represented the changing world state, and a temporal knowledge base agent. These various agents were invoked by the problem solving agent in response to user inputs. A short excerpt from a typical session is shown in Figure 5.

The CAMPS Mission Planner [Emerson and Burstein, 1999] is one of several prototype scheduling tools developed jointly by BBN Technologies, Kestrel Institute and CMU for the US Air Force. It is currently undergoing refinement and integration in preparation for deployment in about a year's time. As part of the DARPA Control of Agent-Based Systems Program, BBN has "agent-ized" the Mission Planner (MP) for experimental use in mixed-initiative multi-agent systems. The MP takes as inputs a set of "requirements," each consisting of quantities of cargo and people to be moved from one airport to another during some time interval. It produces detailed schedules specifying the times at which an aircraft of some type will fly from where they are based to pick up this cargo and carry it to its destination, and then return to home base. Requirement sets can be quite large, numbering hundreds or even thousands of elements, and hundreds of tons of cargo, from tens of locations. Numerous constraints must be satisfied simultaneously to produce valid schedules, which the scheduler can do in seconds or minutes.

A simplistic view of the task faced by a user of the Mission Planner is, given a set of requirements, is to specify a set of suitable aircraft resources, the ports to be made available for refueling (or locations for aerial refueling), should that be necessary, and to ensure that the schedule produced moves all of the requirements by their due dates. The scheduler will fail to schedule flights for all of the cargo requirements if there are constraint violations, or insufficient resources provided, in which case some cargo may be scheduled to arrive late or not at all. As originally

developed, the CAMPS Mission Planner had a traditional graphical user interface for specifying input parameters, and a variety of views of the product schedules produced, including maps, tables and GANTT charts. All user interactions were by keyboard and mouse.

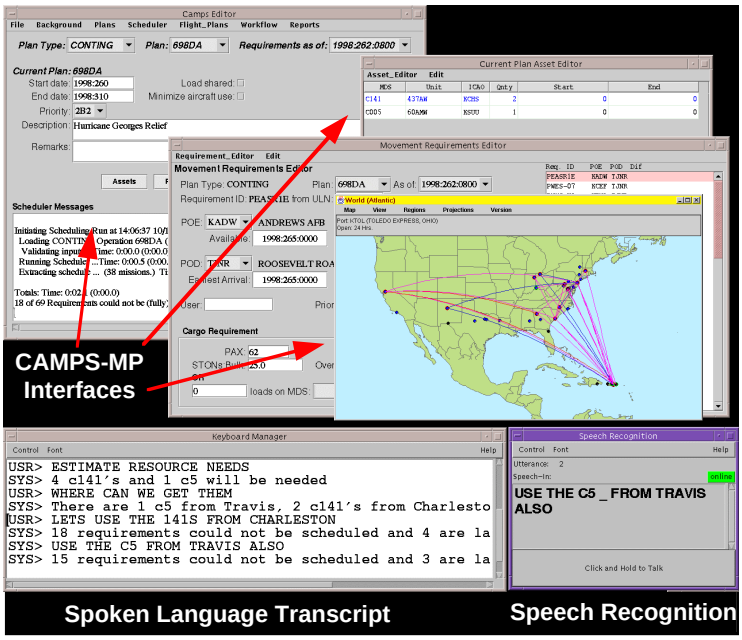
A second scheduling tool, called the **Barrel Allocator** [Becker and Smith, 2000] was developed jointly by CMU and BBN for another community of users (called Barrelmasters for historical reasons) within the same Air Force planning organization. It is used to manage all of the cargo aircraft available from different bases around the country. This tool takes as input the schedules produced by planners once those plans have been committed to, and finds an allocation of aircraft to missions that makes the most productive use of the limited available resources. As more airlift missions are planned by different planners than there are aircraft to fly them all, and because they are often planning far in advance, planners often plan against notional resources, rather than checking for where aircraft will be available. This tool is used to commit particular bases to particular missions, by priority, proximity, and availability, and if necessary reschedule flights to use aircraft from different bases then originally planned. To represent the interaction between the planners and the allocators, a simplified version of the Barrel system was developed into an agent for use in the prototype system.

3 Preliminary Integration Experiment

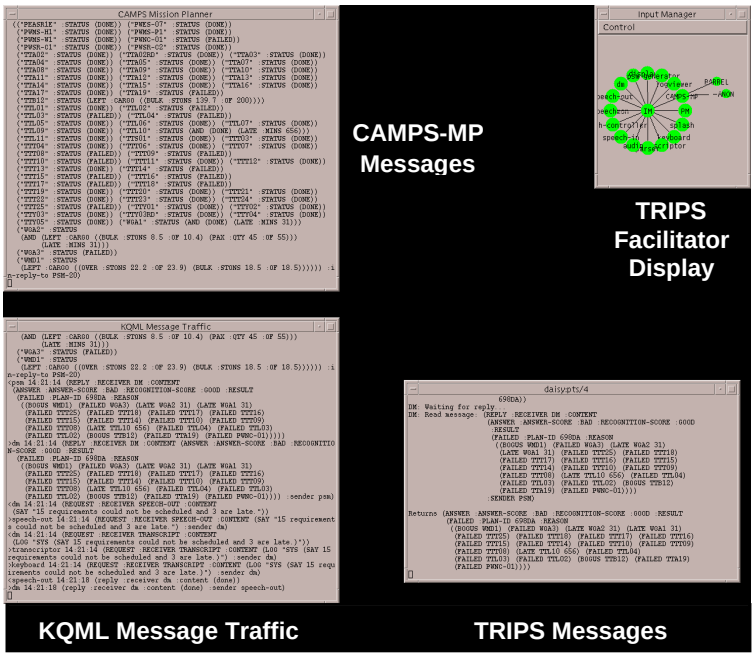
In developing this demonstration system, our approach was to provide a straightforward API to the MP and BARREL agents the KQML agent communications language, and to collect and catalog the reasoning and information interchange issues of the various agents forming the complete system. This included problems arising in interpretation and reformulation of user intent, and the planning of requests to the various back-end, airlift-domain-specific agents.

Figure 4 shows some of the graphical views as seen from a user's perspective, as well as a "behind the curtain" view of the system's activity and inter-agent message traffic. Figure 5 shows a sample dialogue with the system. All of the dialogue was verbal, and the conversation was managed by the TRIPS agents. The graphic displays were generated by the CAMPS-MP agent, using pre-existing views, with one exception. While a user could have accomplished much the same result by interaction with the CAMPS-MP GUIs, many steps would have required tedious entry of information through dialog windows.

The sample dialogue illustrates much of the capability of the combined system at present. Using spoken language dialogue, users can quickly arrive at a solution to typical scheduling problems, in many ways more easily and more naturally than through the GUI. They can select problems to work on, request advice on the



(a) User view



(b) System view

Figure 4: User (a) and system (b) views of TRIPS/CAMPS system

001 U: **Hello.**
002 T: Hey there.
003 U: **Show me the active plans.**
004 T: Ok.
[CAMPS-MP displays a list of plans under development and returns their IDs and descriptions to TRIPS.]
005 U: **Let's work on the Georges plan.**
006 T: Alright.
[CAMPS-MP main viewer focusses on Hurricane Georges deployment plan.]
007 U: **Estimate resource needs.**
[TRIPS requests the estimate from CAMPS-MP.]
008 T: Four C141s and 1 C5 will be needed.
[TRIPS forms its utterance using CAMPS-MP's answer.]
009 U: **Where can we get them?**
[TRIPS sends a set *of requests to the BARREL agent corresponding to the two sets of (nominal) aircraft identified as "them".]*
010 T: There are one C5 from Travis, two C141's from McGuire, and three C141's from Charleston.
[TRIPS integrates the BARREL's replies and formulates the response.]
011 U: **Let's use two 141s from McGuire.**
[TRIPS interprets "use" as a request to provide the given aircraft as resources for scheduling, using the `AUGMENT-PLANSET` *request to CAMPS-MP.]*
012 T: Ok. You still need one C5 and two C141s before we can schedule.
[TRIPS uses the original resource estimate to prompt the user for additional information before invoking the scheduler.]
013 U: **Use the C5 from Travis also.**
014 T: Ok. You still need two C141s before we can schedule.
015 U: **And use the 141s from Charleston.**
[Once TRIPS believes the scheduling problem is sufficiently specified, it requests that CAMPS-MP invoke the scheduler.]
016 T: 12 requirements could not be scheduled.
[CAMPS-MP indicates the status of the plan after scheduling, which TRIPS summarizes for the user.]
017 U: **Show them to me.**
018 T: Ok.
[CAMPS-MP displays the failed requirements in a table view.]
019 U: **Explain PWNC-1.**
["PWNC-1" is the ID of one requirement to move cargo from Washington County Regional Airport to Puerto Rico.]
020 T: C5s can't land at Washington County Regional.

Figure 5: Annotated TRIPS/CAMPS sample dialogue

021 U: **Let's use a C17 instead.**
[This aircraft type can use a shorter runway.]

022 T: One requirement is still unscheduled.
[TRIPS locates a C17 by querying the BARREL, provides it to CAMPS-MP as a resource, then invokes the scheduler again.]

023 U: **Explain it.**
[TRIPS identifies "it" as the failed requirement.]

024 T: C17s, C5s, and C141s cannot fly non-stop from Elmendorf to Roosevelt Roads.
[Elmendorf AFB (source of the requirement) is in Alaska, Roosevelt Roads (destination of the requirement) is in Puerto Rico. The distance between them is too far for the planes to fly without refueling.]

025 U: **Show me a map.**

026 T: Ok.
[TRIPS invokes the CAMPS-MP map viewer with all the routes displayed.]

027 U: **Use Charleston as an en-route stop.**
[TRIPS makes Charleston available to CAMPS-MP as a refueling resource, and reinvokes the scheduler.]

028 T: Ok. All requirements are scheduled.

Figure 5: Annotated TRIPS/CAMPS sample dialogue (cont.)

resources required to solve the problem, suggest particular assets to use, request views of the state of problem definition and of the solutions produced, and receive explanations when things don't go as expected. TRIPS manages the interaction with CAMPS-MP and the BARREL agent, and automatically invokes the scheduler when it is appropriate.

This first-cut integration of TRIPS and CAMPS agents was developed in a relatively short period of time (approximately three months) in part because of the modular, agent-oriented construction of TRIPS itself. Aside from small efforts for extending the vocabulary and language understanding modules, the main effort was in "agent-ifying" the aiflift domain agents, CAMPS-MP and BARREL, and in developing a specialized ProBlem Solving Agent that could do the required interpretation and translation of user actions and airlift agent responses, in the context of the airlift mission planner's task.

The ongoing scientific focus of our effort is now on understanding the capabilities needed in a problem solver that can manage this kind of interaction. In particular, we are now developing a model of the overall problem solving task, that can be used by a more general purpose problem solving agent to interpret user requests in terms of a more declaratively described domain model and relate these to models of the back-end agents.

4 Major Technical Issues

The main focus of this paper is on "lessons learned" about the level of information sharing required of the domain specific agents, and the kinds of reformulation and translation of user requests that is required to facilitate naturalistic, mixed-initiative dialogues with fieldable domain-specific planning/scheduling tools like that shown in Figure 4. Although there are a few other systems, such as Quickset [Cohen *et al.*, 1997] and TRIPS itself, that have demonstrated some of these capabilities, what is new here is the coupling of these largely domain-independent interaction modality-management agents with domain-specific planning systems that were not originally designed for such coordinated, mixed-initiative, multi-agent behavior.

4.1 "Agent-izing" the Mission Planner

The CAMPS Mission Planner system is itself composed of a GUI written in JAVA and a server written in LISP. It was turned into a single agent by providing the LISP server with a KQML interface. We developed a KQML API that provided essentially a programmatic means of invoking the behaviors available through the scheduler's graphical user interface. This was done without explicit reference to the internal representations manipulated by and shared among the TRIPS agents. Table 1 shows a summary of the message signatures handled by the CAMPS-MP agent. It includes a set of commands for manipulating the various views provided by the CAMPS-MP interface, a set of commands for describing new problems and revising old problems to be posed to the scheduler, a request for invoking the scheduler, and queries for analyses of scheduler results and explanations of problems found in those results.

The main information passing messages (`NEW-PLANSET`, `REVISE-PLANSET`, and `AUGMENT-PLANSET`) all take as keyword arguments all of the different kinds of data elements that define problems for the Mission Planner to solve. `NEW-PLANSET` is used to create a new problem from scratch, while `AUGMENT-PLANSET` is used to add ports, resources or requirements to an existing problem. `REVISE-PLANSET` is used to replace one of the previously specified sets of values. We believe that this interface is representative of the kind of interface that one will get in practice when "wrapping" a legacy system. As such we saw it as a reasonable target for the problem solver agent that it was to interact with.

We did add functionality to the CAMPS-MP system in several places to facilitate the kind of dialogue illustrated in Figure 5. The most important of these was that we developed a capability to estimate the resource levels needed to solve the scheduling problem. This capability is now being considered as a requirement for the CAMPS Mission Planner that will be deployed next year (without the natural

KQML Performative	Content Form
REQUEST	(OPEN-VIEWER :VIEWER <VIEWER>)
REQUEST	(FOCUS-VIEWER :VIEWER <VIEWER> ...)
REQUEST	(CLOSE-VIEWER :VIEWER <VIEWER>)
REQUEST	(DISPLAY-TABLE :TITLE <STRING> :QUERY <SQL>)
REQUEST	(MAP-FOCUS :LATITUDE <LAT> :LONGITUDE <LON> :ZOOM <FLOAT> :PLAN-ID <ID>)
INFORM	(NEW-PLANSET :PLAN-ID <ID> :START-DATE <DATE> :END-DATE <DATE> :PRIORITY <PRI> :REQUIREMENTS (<REQUIREMENT>*) :AVAILABLE-ASSETS ((<ACTYPE> <QTY> <UNITID> <START> <END>)*) :AVAILABLE-PORTS ((<LOCID> <NAME> <ENROUTE?> ...)*) ...)
INFORM	(REVISE-PLANSET *keywords same as* NEW-PLANSET)
INFORM	(AUGMENT-PLANSET *keywords same as* NEW-PLANSET)
ASK-ONE	(FLIGHT-PLAN :PLAN-ID <ID>)
ASK-ALL	(LIST-PLANSETS :FIELDS ...)
REQUEST	(SHOW-PLANSETS :PLAN-TYPE <str> :NEW-SINCE <time> :CHANGED-SINCE <time>...)
REQUEST	(ANALYZE-PLAN :PLAN-ID <ID>)
REQUEST	(SHOW-UNSATISFIED-REQUIREMENTS :PLAN-ID <ID>)
REQUEST	(SCHEDULE-AIRLIFT :PLAN-ID <ID>)
REQUEST	(ESTIMATE-RESOURCES :PLAN-ID <ID>)
REQUEST	(EXPLAIN :OBJECT (UNSATISFIED :REQ-ID <ID>))

Table 1: Summary of KQML interface to CAMPS-MP

language interface). We also added several new graphical views to summarize in tabular form queries that seemed natural to ask for in English, such as the view showing the "open problems list" that is requested at the beginning of the sample dialogue.

4.2 Supporting Mixed-Initiative Explanation

Unlike the legacy scheduling system that the CAMPS Mission Planner is being developed to replace, CAMPS-MP included a capability to generate simple explanations of scheduler failures prior to this integration experiment. This was important for the mixed-initiative interactions we sought to demonstrate. CAMPS-MP generates explanations for unscheduled or late requirements by looking for necessary constraints in the problem statement (the scheduler inputs) that make it impossible for the scheduler to successfully schedule flights for all of the requirements. These constraints are those that must be satisfied to form a sequences of flights by an aircraft going from its home base to the cargo to be moved, loading and move the cargo to its destination, and then returning the aircraft to its base. The constraints currently checked for explanation generation are:

- The availability of an aircraft with appropriate carrying capacity for the kind of cargo to be moved. The natural user response to this kind of constraint failure is to introduce a new kind of asset as an available resource in the statement of the problem.

- The capability of an appropriate aircraft to land at all ports along the route. This is checked by comparing the maximum runway length of the port to the minimum runway length required. Again, the appropriate repair is to introduce an aircraft of a type that can land at all ports along the route.

- The capability of the aircraft to fly the distances between each port along its route without refueling. The CAMPS-MP scheduler will ordinarily attempt to land the aircraft at an en-route stop, or use an aerial refueling if either of those options is provided at a location along the route. The repair here is to introduce or designate such a location as part of the scheduler's problem description.

Of course, there are a number of other constraints that can interact to produce scheduling failures (see Emerson and Burstein, 1999). Explaining these more complex failures is an ongoing topic of the CAMPS-MP development effort. Given that this preliminary explanation facility already existed, the main change required for the integration of TRIPS and CAMPS-MP was the production of those explanations in a declarative representation, suitable for inclusion in a KQML message, rather

than directly as "canned" text messages to the user. For the limited set of constraints now handled, this was easily accomplished. Responses to KQML queries of the form

```
(ASK-ALL
  :CONTENT (EXPLAIN :OBJECT (UNSCHEDULED :REQ-ID <ID>))
```

refer to these constraint failures by one of the following:

```
(NOT (CAN-LAND-AT <ac types> <port>)) |
(NOT (CAN-FLY-NONSTOP <ac types> <port1> <port2>)) |
(NOT (CAN-CARRY-CARGO <ac types> <reqid>))
```

and these are in turn translated into English by the TRIPS generator, where their content becomes part of the dialogue context. Ultimately, we would expect that the problem solver could make more direct use of this information, to suggest or even carry out the appropriate repairs, if no further user input is needed to proceed.

4.3 Context Sharing to Support Mixed-Initiative Interaction

Recall that our goal is to provide the user with a seamless view of the agents at their disposal, by making the system itself an intelligent agent that interacts with the user. A crucial issue in supporting this appearance while integrating legacy systems is the need to share contextual information among the agents. Logically, anything that the user sees becomes part of the shared dialogue context, and he or she can refer to it in subsequent communications with the system. In the TRIPS/CAMPS system, this issue arose during the design of the agentized Mission Planner's API because the back-end domain agents hold much of the detailed knowledge of the problem state *and are responsible for displaying that information graphically to the user*. If the system had been engineered from scratch as a single integrated system, so that all user displays were handled by a distinct agent, the problem would have still occurred, but the required communications patterns would have been different.

Consider what happens when the user requests a display of information, such as the list of the active plans needing to be worked on, or a map of the scheduled routes. All of the information displayed in the table generated by CAMPS-MP becomes subject to reference as part of the current dialogue context. The model of the problem solving task we are developing would suggest that the user actions following a request for the list of active plans is either to select one of them to work on, or to ask for more information about one or several of them. In either case, the problem solver, the reference resolution agent, and perhaps even the speech and

language agents require that the list of plans, including their IDs and names, are known and are now salient.

To manage this in our prototype, we used the design principle that all messages requesting that displays be shown return, in addition to an acknowledgment, a declaratively represented summary of the displayed information. Alternatively, one might require that all agents displaying information be capable of being queried for the content of their displays.

The larger issue is how this kind of information is uniformly made available to all of the agents requiring it. We know of no general solution to this. In agent architectures like the star topologies of TRIPS and OAA [Cohen *et al.*, 1994], this information could, in principle, be broadcast to all agents who care to use it. In network communications architectures like RETSINA [Sycara *et al.*, 1996], one might require that some agent or set of agents serve as a blackboard or clearinghouse for the information. Beyond the question of who gets the information, a key remaining problem is that the descriptions of contextual items must include semantic information. For example, the TRIPS components of our system need to know not only that "PWNC-01" is being displayed, but also that it is a transport requirement, that its status is unscheduled, and so on.

4.4 Problem Solving: Plan Recognition, Task Allocation, and Agent Management

Most of the burden involved in actually helping the user with their task falls to the Problem Solving (PS) agent. Its responsibilities in the TRIPS/CAMPS architecture can be divided into three levels.

First, it is responsible for interacting with the Conversational Agent when the latter determines that the interpretation of the current utterance involves domain-level problem-solving. At this level, the Problem Solver checks whether the proposed interpretation is plausible in the current problem-solving context. In general, this requires some form of plan recognition. Although for the simplified airlift mission planner's task used in our integration experiment, simple heuristics were sufficient, we are now building a more complete model of the task suitable for use in this interpretation phase.

Second, if the interpretation seems to make sense, the Problem Solver must determine how to meet its obligation to respond to it. This might involve modifying the problem solving state, as in a suggestion to extend or modify the parameters of the problem (*e.g.*, "Use the C5 from Charleston also"), or determining the answer to a query (*e.g.*, "Where can we get them [the assets]?"), and so on. At this level, the Problem Solver needs to use its knowledge of the problem solving state and of the capabilities of the agents at its disposal to produce a plan for using those agents

to fulfill its obligations. In most cases, this requires matching agent capabilities against outstanding obligations, using some general principles for decomposing tasks into pieces those agents can handle.

An interesting special case here is interactive plan repair. When the result of the previous actions by the user and/or the system is the discovery of a problem with the partial plan produced, the user may ask for (or, better yet, the system may generate on its own) an explanation of the failure. The explanation sets the context for some particular class(es) of repairs, as described in Section 4.2 above. In the sample dialogue with the present system, essentially context-independent interpretation of the user response "Use a C17 instead" allows the planning to proceed, but a seemingly subtle change to "Use a C17" would be ambiguous, and the system could not proceed. Without a model of the plausible repairs for the failure, the system would not be able to prefer the substitution interpretation over the addition of additional assets to the existing specification.

Finally, at the third level, the Problem Solver must execute this plan and actually interact with the agents to obtain the information needed to fulfill its obligations. This involves translating between the general-purpose representation used by the Problem Solver and whatever API is available from the agent being invoked. Typically, multiple messages must be sent, since the Problem Solver's obligations come from the expressive language interface, whereas individual agent's APIs are usually quite simple. An example here is the interaction with the BARREL agent to find available assets ("Where can we get them?"). The Problem Solver, having resolved the reference to the list of notional assets estimated by CAMPS-MP to be needed, must generate a sequence of queries to the BARREL to determine where each type of asset can be found. The replies must be gathered and interpreted, and the Problem Solver moves on to the next step in its plan (barring an unexpected response or error indication, which would abort the plan or trigger other remedial action, such as entering into a sub-dialogue with the user).

Once all the necessary information has been obtained, a suitable response must be produced for use by the Conversational Agent in maintaining the dialogue context and by the realization components for presentation to the user. The Problem Solver returns both the plausibility estimate for each interpretation of the user's intent, as determined in Step 1, and an answer score indicating the quality of the solution found in Steps 2 and 3. The Conversational Agent uses both of these values to select the preferred interpretation of the user's utterance and produce an appropriate response. When the response requires use of another agent's display capabilities (as with CAMPS displaying a map or table), the Problem Solver manages the invocation of the other agent(s), with suitable translation and acknowledgments.

In the initial demonstration system, Steps 2 and 3 of this process are not performed as generally as described above. Instead, the Problem Solver has a single

model of the airlift mission planner's task together with how the CAMPS-MP and BARREL agents fit into that task. We are now developing a framework to support declarative representation of the user's problem solving task and of the capabilities of the back-end agents, thereby permitting a more direct and general implementation of the problem solving model described above. One of the results of this work will be a system that can more easily be used in a variety of mixed-initiative tasks and with a dynamically changing set of back-end agents.

5 Conclusions and Directions

We have described an initial integration of the TRIPS agent framework for multi-modal mixed-initiative control of planning support agents with several domain-specific agents representing aspects of the airlift planning domain that were not originally designed to work in such a framework. Our larger objectives here are to develop the framework to support appropriate utilization of a wider variety of agents in support of user objectives, and to utilize those agents more proactively when the opportunities arise.

In the process, we have uncovered, or, in some cases, rediscovered, requirements for both the problem solver mediating between the user and these domain agents, and for the capabilities of the domain agents that will be suitable for participating in such systems. In particular, for "legacy agents," we have discussed the importance of information sharing between agents that can present information to the user and the dialogue components, and the important role that a capability to generate explanations can play in furthering the cooperative problem solving of the user+agents team.

For the problem solver, we have again motivated the need for both capability as well as *use* models for the agents it is directing. We believe that these models will be utilized primarily in a reactive planning framework, with some outstanding issues. First, the translation of user requests into subtasks for other agents in general requires some amount of reformulation and decomposition to match the API of the target agents. In part, this is a generative planning task not unlike that found in systems for distributed information retrieval like SIMS and ARIADNE [Arens *et al.*, 1996; Ambite and Knoblock, 1997] It could perhaps be largely decoupled from the problem solver's role in context-sensitive intent interpretation and subgoal formation.

Second, and perhaps more problematic, is the development of a general framework for the Problem Solver's role in multi-modal response generation. Here, to be cooperative in a mixed-initiative sense, the problem solver must be capable of summarization (as when reporting the results of scheduling by characterizing the

failures), and also of directing the user to view salient visualizations, when they are available through some agent at its disposal.

Finally, we see the development of task/use models in the problem solver as providing excellent opportunities for improving the initiative of such systems. At present, TRIPS/CAMPS takes initiative simply when it has enough information to attempt calling the scheduler (based on a description of the scheduling capability). A use model could predict when discovered failures should lead directly to an (unprompted) request of MP for an explanation, and even potentially to routine repairs, when no further discrimination is required of the user. These models could also perhaps be the locus for machine learning about appropriate times for initiative, if a system like TRIPS/CAMPS was routinely used to solve similar kinds of scheduling problems, as would undoubtedly be the case in the airlift domain.

References

[Allen *et al.*, 1995] James F. Allen, Lenhart K. Schubert, George Ferguson, Peter Heeman, Chung Hee Hwang, Tsuneaki Kato, Marc Light, Nathaniel G. Martin, Bradford W. Miller, Massimo Poesio, and David R. Traum, "The TRAINS Project: A case study in defining a conversational planning agent," *Journal of Experimental and Theoretical AI*, 7:7–48, 1995.

[Ambite and Knoblock, 1997] Jose-Luis Ambite and Craig A. Knoblock, "Planning by rewriting: Efficiently generating high-quality plans," In *Proceedings of the Fourteenth National Conference on Artifical Intelligence (AAAI-97)*, 1997.

[Arens *et al.*, 1996] Yigal Arens, Craig A. Knoblock, and Wei-Min Shen, "Query reformulation for dynamic information integration," *Journal of Intelligent Information Systems*, 6(2/3):33–130, 1996.

[Becker and Smith, 2000] Marcel A. Becker and Stephen F. Smith, "Mixed-Initiative Resource Management: The AMC Barrel Allocator," In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning and Scheduling (AIPS-2000)*, Breckenridge, CO, April 2000.

[Burstein and McDermott, 1996] Mark H. Burstein and Drew McDermott, "Issues in the Development of Human-Computer Mixed-Initiative Planning," In B. Gorayska and J.L. Mey, editors, *Cognitive Technology*, pages 285–303. Elsevier, 1996.

[Burstein *et al.*, 1998] Mark H. Burstein, Alice M. Mulvehill, and Steve Deutsch, "An Approach to Mixed-Initiative Management of Heterogeneous Software Agent Teams.," In *Proceedings of the Thirty-first Hawaii International Conference on the System Sciences (HICCS-98)*, Kona, HI, 6-9 January 1998.

[Cohen *et al.*, 1994] Philip R. Cohen, Adam J. Cheyer, M. Wang, and S. C. Baeg, "An Open Agent Architecture," In *AAAI Spring Symposium on Software Agents*, pages 1–8, March 1994.

[Cohen *et al.*, 1997] Philip R. Cohen, Michael Johnston, David McGee, Sharon Oviatt, Jay Pittman, Ira Smith, Liang Chen, and Josh Clow, "Quickset: Multimodal interaction for distributed applications," In *Proceedings of the Fifth Annual International Multimodal Conference (Multimedia-97)*, pages 31–40. ACM Press, 1997.

[Emerson and Burstein, 1999] T. Emerson and Mark Burstein, "Development of a Constraint-based Airlift Scheduler by Program Synthesis from Formal Specifications," In *Proceedings of the Conference on Automated Software Engineering*. AAAI Press, 1999.

[Ferguson *et al.*, 1996a] George Ferguson, James Allen, and Brad Miller, "TRAINS-95: Towards a Mixed-Initiative Planning Assistant," In Brian Drabble, editor, *Proceedings of the Third Conference on Artificial Intelligence Planning Systems (AIPS-96)*, pages 70–77, Edinburgh, Scotland, 29–31 May 1996.

[Ferguson and Allen, 1998] George Ferguson and James F. Allen, "TRIPS: An Integrated Intelligent Problem-Solving Assistant," In *Proceedings of the Fifteenth National Conference on AI (AAAI-98)*, pages 567–573, Madison, WI, 26–30 July 1998.

[Ferguson *et al.*, 1996b] George Ferguson, James F. Allen, Brad W. Miller, and Eric K. Ringger, "The Design and Implementation of the TRAINS-96 System: A Prototype Mixed-Initiative Planning Assistant," TRAINS Technical Note 96-5, Department of Computer Science, University of Rochester, Rochester, NY, October 1996.

[Finin *et al.*, 1993] Tim Finin, Jay Weber, Gio Wiederhold, Michael Genesereth, Richard Fritzson, Donald McKay, James McGuire, Richard Pelavin, Stuart Shapiro, and Chris Beck, "Specification of the KQML Agent-Communication Language," Draft, 15 June 1993.

[Labrou and Finin, 1997] Yannis Labrou and Tim Finin, "A Proposal for a new KQML Specification," Technical Report CS-97-03, Computer Science and Electrical Engineering Department, University of Maryland Baltimore County, Baltimore, MD, February 1997.

[Sycara *et al.*, 1996] Katia Sycara, Keith Decker, Amandeep Pannu, Mike Williamson, and Dajun Zeng, "Distributed Intelligent Agents," *IEEE Expert*, December 1996.