Conversational Case-Based Planning for Agent Team Coordination

Joseph A. Giampapa and Katia Sycara

The Robotics Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213–3890 (U.S.A.)
{garof, katia}+@cs.cmu.edu
http://www.cs.cmu.edu/~{garof, katia, softagents}

Abstract. This paper describes a prototype in which a conversational case-based reasoner, NaCoDAE, was agentified and inserted in the RETSINA multi-agent system. Its task was to determine agent roles within a heterogeneous society of agents, where the agents may use capability-based or team-oriented agent coordination strategies. There were three reasons for assigning this task to NaCoDAE: (1) to relieve the agents of the overhead of determining, for themselves, if they should be involved in the task, or not; (2) to convert seemingly unrelated data into contextually relevant knowledge — as a case-based reasoning system, NaCoDAE is particularly suited for applying apparently incoherent data to a wide variety of domain-specific situations; and (3) as a conversational CBR system, to both unobtrusively listen to human statements and to proactively dialogue with other agents in a more goal-directed approach to gathering relevant information. The cases maintained by NaCoDAE have *question* and *answer* components, which were originally intended to maintain the textual representations of questions and answers for humans. By associating agent capability descriptions and queries with the case questions, NaCoDAE also assumed the team role of a capability-based coordinator. By encoding fragments of HTN plan objectives in its case actions, we were able to convert NaCoDAE into a conversational case-based planner that served compositionally-generated HTN plan objectives, already populated with situation-relevant knowledge, for use by the RETSINA team-oriented agents.[1]

[1] The authors are grateful to the Naval Research Labs for providing the sources to NaCoDAE. Matthew W. Easterday made a significant contribution to this project by adapting NaCoDAE to operate in an agent context. Many thanks to Alex Rudnicky for allowing us to agentify Sphinx and for providing us with technical support. On a personal note, Joseph Giampapa would like to thank David Aha for his encouragement and helpful suggestions. This research was sponsored in part by the Office of Naval Research Grant N-00014-96-16-1-1222 and by DARPA Grant F-30602-98-2-0138.

1 Introduction

Complex tasks are often solved by teams because no one individual has the collective expertise, information, or resources required for effective performance. Team problem solving involves a multitude of activities such as gathering, interpreting and exchanging information, creating and identifying alternative courses of action, choosing among alternatives by becoming aware of differing and often conflicting preferences of action by team members, and ultimately implementing a choice, determining how incremental progress will be measured, and monitoring its evolution. There have been many attempts and views in the agent community as to how intelligent software agents should organize themselves into teams. The work of this paper was inspired by the *joint intentions* theory of Cohen and Levesque [5, 14], the *shared plans* theory proposed by Barbara Grosz [12], and Milind Tambe's research based on the *TEAMCORE* [19, 28] multi-agent software system.

In the joint intentions theory, a team is composed of agents that jointly commit to the achievement of a *joint persistent goal*, or JPG. Agents desiring to be part of a team must communicate their intention to each other that they intend to commit to that goal. A team is formed once every agent has committed to a goal and has received a communication that all the other agents have committed to it, as well. The team remains a team as long as: (a) no agent has a reason to believe that the goal is unachievable; (b) no agent decides, on its own, to decommit from the team goal; (c) the goal is not yet achieved and all agents remain convinced that it is still achievable; and (d) no agent perceives that another agent has de-committed from the team goal. Team goals are formed by an individual agent nominating a task as a proposed team goal, and communicating that intention until consensus is formed that the nominee is worth pursuing as a team goal. The joint intentions theory is significant because it is a formal model of what motivates agent communications about teamwork. Further, it has enjoyed broad recognition within the agent community as making pertinent claims and observations about team-oriented behaviors. But the theory does not address: (1) the problems of how agents acquire a team goal; (2) how agents identify the roles that contribute to the fulfillment of a team goal, and identify those roles for themselves; (3) how agents relate the roles of their individual goals to the overall goal of the team; and (4) how the agents know when to break commitments to their individual goals while still maintaining the team goal.

The shared plan theory [12] emphasizes the need for a common high-level team model that allows agents to understand all requirements for plans that might achieve a team goal, even if the individuals may not know the specific details of the plans or how the requirements will be met. This allows team members to map their capabilities to a plan to achieve a team goal, assign roles to themselves, and measure their progress at achieving their overall team objective. Like joint intentions theory, shared plan theory is based on observations of human forms of teamwork.

TEAMCORE is an agent architecture that implements many of the basic principles of joint intention theory. TEAMCORE application scenarios are usu-

ally situated in the military and robotic soccer domains, where the team-oriented agents are homogeneous and their roles typically represent either authority relations, such as military rank, or high-level capability descriptions, such as *transport* or *escort* helicopters. But once individual TEAMCORE agents commit to being part of a team, they cannot dynamically add or subtract members to or from their team so as to adapt to a new situation, while executing their plan.

Within human-machine teams, intelligent software agents can play a variety of roles that help reduce some of the overhead of teamwork, as well as help solve team problems. By means of their autonomy, agents can: get information requested by a human; self-activate and present unsolicited important information to a human user; suggest solutions to a problem; actively monitor the environment and cache relevant information so as to provide quick updates to "situation knowledge" if required; and recombine, as needed, with other agents to adapt to the particular task requirements over time.

To coordinate agents into teams effectively in dynamic environments, our research thrust has been in line with the following principles: (1) we make open world assumptions about the nature of tasks and the strategies for solving them — a multi-agent solution to a problem will most likely involve agents of different architectures and abilities, available at different times; (2) we subscribe to the belief that there are meta rules that describe the nature of teamwork, that are independent of the specific task being performed [8, 18, 21], and that it is possible to reuse this knowledge in different application domains; and (3) that individual roles and objectives of agents within the team may need to change in order to maintain and achieve the full team goal.

In this paper we describe a prototype, implemented in our RETSINA[2] multi-agent infrastructure, in which agents interact with each other via capability-based [23] and team-oriented coordination. For the team-oriented agent coordination to be effective, we propose a model of teamwork based on the joint intentions theory for agent communications about their intended commitments, combined with the shared plans strategy of expressing descriptions of roles and context-specific requirements for those roles. We enhance this proposal by adding our own characterizations of role and subgoal relations in software agent teamwork, and show how the software agents can acquire this information from their operating environment during execution time. The acquisition and maintenance of the contextual information that determines the plan requirements is performed by NaCoDAE, a conversational case-based reasoner, which is used to compositionally generate Hierarchical Task Network [HTN] plan [9, 16] objectives for the RETSINA team agents. We show that the unobtrusive and invisible use of NaCoDAE as the primary means by which human and agent information is gathered and merged can eliminate any information overload that might result from the conscious interactions of humans with their intelligent agents.

In the sections that follow, we present RETSINA, NaCoDAE, the interacting NaCoDAE and RETSINA prototype, and a command and control scenario that we used as a case study for NaCoDAE's effectiveness as an agentified conversa-

[2] **R**eusable **E**nvironment for **T**ask-**S**tructured **I**ntelligent **N**etwork **A**gents

tional case-based planner. After a brief review of related work, we conclude with some ideas for future work in this area.

2 The RETSINA Multi-Agent System

The RETSINA multi-agent system (MAS) is a collection of heterogeneous software entities that collaborate with each other to either provide a result or service to other software entities or to an end user. As a society, RETSINA agents can be described in terms of the *RETSINA Functional Architecture* [27, 24], illustrated by Figure 1, which categorizes agents as belonging to any of four agent types:

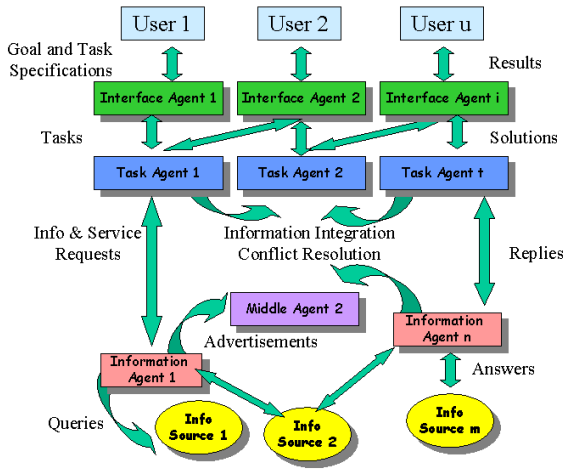

Fig. 1. The RETSINA Functional Architecture

Interface agents present agent results to the user, or solicit input from the user. In addition, they could learn from user actions [4]. Interface agents typically represent specific modes of input or output, such as a *VoiceRecognition* agent or a *SpeechGeneration* agent, or can be associated with different device types such as *PDA*, *Phone*, or *E-Mail* agents. Interface agent behaviors can also be associated with *task agents*.

Task agents encapsulate task-specific knowledge and use that knowledge as the criterion for requesting or performing services for other agents or humans. In this respect, they are the typical agent coordinators of a multi-agent system.

Middle agents [29, 10] provide infrastructure for other agents. A typical instance of a middle agent is the *Matchmaker* [25, 26], or *Yellow Pages* agent. Requesting agents submit a *capability* request to the Matchmaker, which will then locate the appropriate service-providing agents based upon their published *capability descriptions*, known as *advertisements*.

Information agents model the information world to the agent society, and can monitor any data- or event-producing source for user-supplied conditions. Information agents may be *single source* if they only model one information source, or may be *multi-source* if one information agent represents multiple information sources. Information agents can also update external data stores, such as databases, if appropriate.

By classifying agents functionally, we believe that it is possible to uniformly define agent behaviors [6] that are consistent with their functional description. For example, information agents implement four behaviors for interacting with the data sources that they model: *ask once*, *monitor actively*, *monitor passively*, and *update*. RETSINA agents typically use the capability-based coordination [23] technique to task each other, which means that one agent will dynamically discover and interact with other agents based on their capability descriptions. RETSINA agents also support other forms of coordination techniques, such as the team-oriented coordination that is described later in this paper.

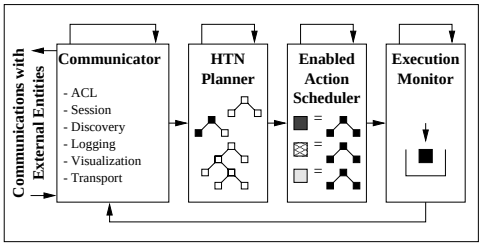

Fig. 2. Schematic diagram of the RETSINA Agent Architecture. The boxes represent concurrent threads and the arrows represent control and data flow. The "external entities" may be agent or non-agent software components.

The *RETSINA Individual Agent Architecture* [24, 3, 6] is illustrated by Figure 2. This agent architecture implements Hierarchical Task Network (HTN) Planning [9, 16] in three parallel execution threads. A fourth thread, the *Communicator* [20], provides the means by which the agent communicates with the networked world. The Communicator provides a level of abstraction that insulates the planning component from issues of agent communication language (ACL), communication session management, the location of agent services via discovery, the logging and visualization of agent messages and state information, and the communication transport being used (e.g. infrared, telephone, base band, etc.). The *HTN Planner* thread receives HTN plan *objectives* from the Communicator, extracts the information and instructions contained therein, and attempts to apply the extracted data to all the plans in its plan library. Plan actions are partially *enabled* as the data is applied to them, and once all actions of a plan are completely enabled, they are scheduled by the *Scheduler*. The Scheduler maintains the enabled actions in a priority queue, and works with the *Execution*

Monitor, which actually executes the enabled actions. The coordination among the three planning modules is done in such a way that high-priority actions can interrupt those being executed by the Execution Monitor, if those being executed are of a lower priority.

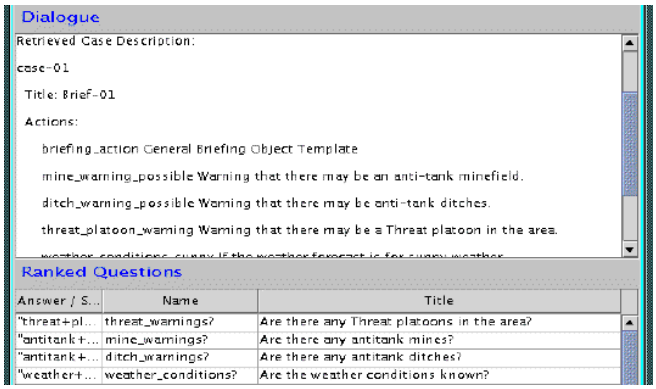

Fig. 3. Case and question areas of the NaCoDAE GUI

3 Navy Conversational Decision Aiding Environment

NaCoDAE [2] is a *conversational case-based reasoning* (CCBR) system that helps a user decide a course of action by engaging him in a dialogue in which he must describe the problem or situation. A conversational session begins with the user providing an initial partial description of the problem that he is trying to solve. NaCoDAE responds by providing a ranked *solution* display, which lists the solutions of stored cases whose problem descriptions best match the user's problem descriptions, and a ranked *question* display, which lists the unaswered questions in these cases. The user interacts with these displays, either refining their problem description by answering select questions, or by directly selecting a solution to apply. By presenting questions in a ranked list, NaCoDAE attempts to help guide the user to a rapid description of the problem by asking what it perceives to be the most relevant questions, given the information provided.

Figure 3 illustrates the way in which NaCoDAE presents its solutions to the user. It displays the case as a bundle of *actions* to take, a textual description of what the case means, a list of the questions, their responses, and the case that resulted from their selection. To make case authoring easier, NaCoDAE also contains a module named *CLiRe* [1][3], that refines case libraries so as to enforce specific case authoring guidelines. CLiRe uses a machine learning technique to refine case libraries.

[3] **C**ase **Li**brary **Re**visor

There were three features of NaCoDAE that made it suitable for team coordination and interaction with RETSINA agents. First, NaCoDAE can work with partial descriptions of the problem and use them for initiating a dialogue. This could allow one to encode a general strategy of, "always knowing the strategy for how to get more information, if nothing else is known" — a technique inspired by one of Barbara Grosz's motivations for shared plans [12]. Second, NaCoDAE can continually revise its list of most likely candidate cases, as data is provided to the system by either an agent or the user. This feature lends itself to a form of coherent, compositional and incremental construction of knowledge structures, such as HTN plan objectives and representations of situational or contextual knowledge. This knowledge can be accessed even if time and the lack of specific information do not allow for a description to be completely specified. Third, the cases can be modified to store any type of textual data, including agent capabilities and queries.

4 The RETSINA Model of Teamwork

Teamwork must be motivated by an overall description of a goal that merits solution by a team, and a description of a shared plan that can achieve the goal. As permitted by the shared plan theory, the plans can be fully or partially specified. A team-oriented individual must possess three types of knowledge: his capabilities, the team plan requirements, and social parameters for role assessment, such as knowledge of his authority to address team plan requirements, and knowledge of social structure, such as superior, peer, and subordinate relationships. To be able to act on that knowledge, a team-oriented individual should also know how to perform certain types of assessments, such as: how to match his individual capabilities to plan requirements; how to evaluate if his authority allows him to apply his capabilities to the plan requirements; how to assess the impact of his and his teammates' roles on achieving the overall team goal, so that he may offer more appropriate role proposals when situations change; how to monitor progress when executing a plan; and how to map social structure to plan requirements, such as knowing to report to an immediate superior or that only particular team members have the authority to assign certain tasks.

The acquisition of situation-specific knowledge serves to "update" an individual's beliefs about the three types of team knowledge, mentioned above. Individuals should attempt to become aware of as much situation-specific knowledge as is necessary, both in the stage of forming a team and committing to roles in the team plan, and while executing the team plan. Some situation-specific knowledge may fill gaps in the partial shared plan and transform it into a full shared plan. Other situation-specific knowledge may modify the individual's knowledge of his capability, social status, or of his authority. Still, another form of situation-specific knowledge is that which is communicated by teammates at consensually-determined *checkpoints* so as to indicate individual progress in relation to the overall team plan.

As requirements of the task change, subgoals are achieved, or as individuals change their capabilities, individuals must communicate these changes to the appropriate teammates. These communications have been found to be critical in human high performance teams [8, 18, 21]. If an individual discovers that he is no longer capable of performing a role, then he must communicate this knowledge to his teammates and superiors (if appropriate) and break with his subgoal commitment. Alternatively, the individual may opt to stay with the subgoal commitment if it does not impede progress to achieving the overall team goal and there is no reason or request to assume another role.

An individual determines candidate roles for himself by matching his individual capabilities to the requirements of the overall team goal within the constraints of his authority and other social parameters.[4] If there are no candidate roles, then the individual has the following options: (1) to attempt to further refine the requirements of the overall team plan; (2) to attempt to acquire those capabilities that match the plan requirements; (3) to attempt to acquire the authority for applying the capability to the plan requirements; and (4) if it is not possible to either specify the requirements, acquire the capability or the authority to generate a candidate role in the team plan, then the individual should not commit to the team plan. If the individual were successful at generating candidate roles, however, then the individual should select the candidate roles that he feels comfortable with committing to — by whatever evaluation metric at his disposition (e.g. most commitments, least commitments, those that are the best for a certain metric, etc.) — and communicate them to his teammates.

An individual with candidate roles must communicate them to the other team members as proposals for his role in the team plan. Similarly, the individual must receive the proposals for roles of the other team members, and evaluate if all plan requirements are covered by all the proposals that were generated or received by the individual. If all role proposals cover all plan requirements without conflicts, then the individual may commit to the team plan and to his roles. If there are no role proposal conflicts but not all plan requirements are met, then the individual must evaluate if the requirements must be met as a precondition to executing the plan. If they are, then the individual should reconsider if he has the capability for addressing the non-assigned plan requirement, since he might have withheld proposing the role for cost reasons. If he does have the capability, and the benefit of achieving the team goal outweighs the cost of committing to that role, then the individual should propose it. If he does not have a capability to respond to the requirement, then he must wait until all other teammates have attempted to bid on it. If the requirement is eventually covered, then the team can commit to the shared plan. If the requirement is not covered, then the team members cannot commit to the shared plan, but they may actively recruit new team members that could cover the requirement.

[4] Since individuals may be committed to roles for the entire duration of the full team plan, or for less time, we often call roles, *subgoals*, as well, and use the two terms interchangeably.

If there are any conflicts, then only those agents with conflicting role proposals must renegotiate their role proposals in a generate-and-repropose cycle. If the conflicting parties cannot resolve their differences, they should enlarge the circle of participants to include non-conflicting individuals, in the hope that new members of the conflict resolution group may have capabilities that can permit a reassigning of proposed roles so as to avoid the conflict. Once conflicts have been resolved by the conflicted individuals, the proposed roles must be recommunicated to all team members to form a shared mental model of the team plan, so that all members can commit to it.

5 The Interaction of NaCoDAE in RETSINA

We placed the agentified version of NaCoDAE, now a RETSINA task agent, in a group of other RETSINA agents that had to organize themselves so as to perform a mission in the *ModSAF* simulation environment [7][5]. There were other agents in that community, but the group that was relevant to the mission was composed of:

BriefingAgent a task agent that, together with NaCoDAE, maintains the domain-specific knowledge of the full and partial shared plans for the MissionAgents, and performs agent-to-agent conversations on behalf of NaCoDAE. It also assembles shared plans from the actions of NaCoDAE cases, and finds MissionAgents to execute these plans by querying the Matchmakers with the platoon capability descriptions as preferences.

DemoDisplay an interface agent that monitors the MessageLogger so as to provide visualization of agent-to-agent communications for humans monitoring the agent system.

Matchmakers there are two types of matchmakers, Gin [25] and LARKS [26], that are capable of different forms of semantic matching. A matchmaker is a middle agent that enables agents to find each other based on their capabilities.

MessageLogger an information agent to which agents send copies of the messages they send to other agents.

MineSweepingTeam a team of task agents that use their own team-oriented coordination strategies to clear a path through a minefield as quickly as possible. [22]

MissionAgents three team-oriented task agents that must plan their joint mission with each other. Each one monitors and commands a platoon on behalf of the human platoon commanders, in the ModSAF simulation environment.

ModSAF_Proxy a task agent that models ModSAF behaviors to the other RETSINA agents, and allows those agents to interact with the ModSAF environment.

MokSAF / PalmSAF three interface agents, installed on three different portable hardware platforms, such as pen tablets or PDAs, that present human users

[5] ModSAF is an acronym for **Mod**ular **S**emi-**A**utomated **F**orces.

with shared plan proposals for team coordination. The proposals show the coordinated planned routes on a ModSAF map. They also solicit the commanders' approval or rejection of the proposed team plans. [17] PalmSAF is a version of MokSAF for the PalmPilot^{TM}.

NaCoDAE a task agent that, together with the BriefingAgent, maintains the knowledge that a superior commanding officer is likely to impart and require from the three platoon commanders. It merges information that is provided by both humans and agents to compositionally generate HTN plan objectives that the plan shared by the MissionAgents aims to fulfill.

NarratorAgent a special type of task agent that, given the number of SpeechAgents in audible range of each other, assigns them different voices, and paces the tasking of SpeechAgents so that their speech does not interfere with each other.

RoutePlanningAgents three task agents, one dedicated to each MissionAgent, that plots routes for the platoons given characteristics of the terrain, the vehicle composition of the platoon, and constraints imposed by the mission and human commanders, such as the need for some routes to be mutually reinforcing. [17]

SpeechAgents three speech generation interface agents that synthesize audible speech from text provided by subscriber agents.

VisualReconnaissance an information agent that scans the ModSAF map and notifies its subscribers of the location of a Threat Platoon when it finds one.

VoiceAgent a voice recognition interface agent that is based on the Sphinx [13] speech recognizer. The VoiceAgent provides a dictation service to any agent that subscribes to it. The service provides the subscriber with a textual representation of what it recognizes from the human user's speech.

WeatherAgents three information agents that permit requesting agents to learn about the current weather conditions from the web sites of: USA Today, CNN, and Intellicast.

Figure 4 illustrates the types of agent communications that involve the BriefingAgent and NaCoDAE. The human figures of the scenario are represented by the irregular hexagonal shapes at the top of the diagram, and their communications by the dashed line, (1). As the Company Commander speaks, his speech is translated into text by his VoiceAgent. The BriefingAgent receives those textual translations, as represented by line (2), and attempts to match the text of the Commander's speech with the textual answers to questions that were posed by NaCoDAE.[6] If there is a match, then the BriefingAgent will send that answer to NaCoDAE, as shown by (3). If NaCoDAE can use that answer to complete a case, then it will return a case to the BriefingAgent (3), otherwise return a regenerated ranked list of questions and their associated answers (3). If NaCoDAE's questions contain agent queries, the BriefingAgent will directly query the provider agent if it is known (5), or first ask either or both of the Matchmakers

[6] The BriefingAgent uses a token subset matching algorithm to permit matches despite variations in word order and phrase length.

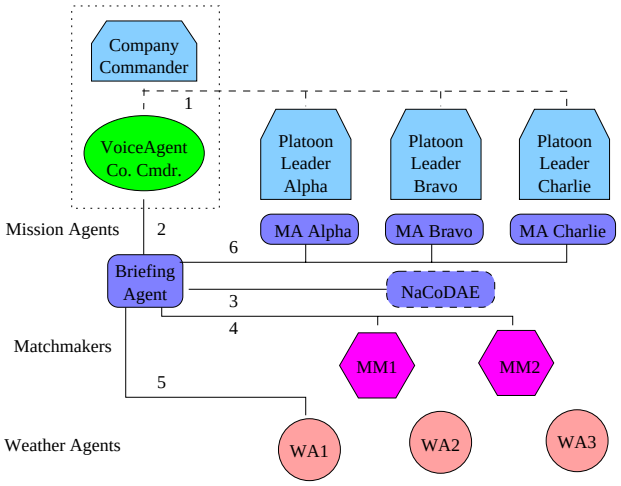

Fig. 4. Agent communications that involved the *BriefingAgent* and *NaCoDAE*

for the identity of a provider agent (4), and then contact it (5). Upon request of the MissionAgents (6), or upon the completion of a case by NaCoDAE (3), the BriefingAgent will assemble a shared plan from the case actions and send it to the MissionAgents (6). During the execution of the scenario, the MissionAgents may also provide the BriefingAgent with updates to their capabilities (6), which the BriefingAgent can forward to NaCoDAE (3).

6 Description of the Scenario

In the scenario, our model of teamwork is implemented by the MissionAgents, knowledge of the shared plans is maintained by NaCoDAE and the BriefingAgent, and the shared plans are represented by *briefing_cases* that contain *briefing_actions* similar to the one in Figure 5. Information that should be part of the briefing is stored in the case-base: map data, additional resources allocated to the team such as mine sweepers and intelligence reports, warnings, information on the adversary, and reminders for the type of information to monitor during the execution of the plan. In the descriptions that follow, references to shared plan content that is represented in Figure 5 will appear in a different font.

The scenario begins with a Company Commander briefing three human platoon leaders. As he speaks, the BriefingAgent eavesdrops on the Commander's discourse via the VoiceAgent. When he describes the composition of the platoons, the BriefingAgent matches that information to some of the anticipated answers to questions that NaCoDAE had ranked as highly likely, and passes that information to NaCoDAE, which then selects cases with descriptions of platoons having the same composition. NaCoDAE sends a revised ranked question list to the BriefingAgent: *Will the platoons have scouts? Is this a nighttime mission? What*

```
BEGIN ACTION briefing_action                                    TITLE "Briefing Object"
TEXT "(briefing_object :brief_type captains-orders
  :goal (goal :type (1 2 primary-goal)
    :1 (description :do move :what your-platoons :to horizontal_line_500)
    :2 (description :do force :what Threat_Platoon :to (behind :landmark
                                                  horizontal_line_500) :mode if-found)
    :primary-goal both)
  :map-checkpoints
    (map-checkpoints :type (Xcoords-Ohio Ycoords-Ohio width-Ohio
      Xcoords-Texas Ycoords-Texas width-Texas
      Xcoords-Utah Ycoords-Utah width-Utah
      checkpoint-1    [...]    checkpoint-11    map-reference)                    [...]
  :plan-requirements (requirements :type (names ohio texas utah distribution)
    :names (ohio texas utah)                    :ohio (maneuverability :rating 3-6)
    :texas (maneuverability :rating 6-8)        :utah (maneuverability :rating 3-6)
    :distribution (description :quantity 1 :assigned negotiate)          )
  :team-capability (team :type (capability-1 capability-2 capability-3)
    :capability-1 (description :maneuverability 5 :firepower 6  [...]          )
  :warning ()    )"
TOOLBOOK                                                        END ACTION
```

Fig. 5. Part of the *Briefing Object*, encoded as a NaCoDAE case action

will the weather conditions be? Are there any known anti-tank mines? etc. Since the question about the weather also contains a query for agents, the BriefingAgent queries the Matchmakers for WeatherAgents, finds some, and queries them for the weather conditions, while the Commander continues with his description of the mission. The WeatherAgents reply "rainy", and NaCoDAE immediately refines its list of solution cases to those which include the action representing the constraint, *plan platoon routes to avoid soft soil areas.*

At the conclusion of the Commander's briefing, the BriefingAgent finds the MissionAgents by querying the Matchmakers for agents with knowledge of team coordination for the command of platoons in ModSAF. Upon receipt of a list of four MissionAgents, the BriefingAgent selects three and then assigns one platoon to each of them. All three MissionAgents receive the same description of the task that requires teamwork in the form of an HTN plan objective, with the briefing_object as the data segment. The goals are: (a) to scout the terrain up to landmark `horizontal_line_500`; (b) to force `Threat_Platoon`, if encountered, behind the landmark; and (c) the team goals are conjunctive: `:primary-goal both`.

A team member's capabilities are those of the platoon that it represents. Since the platoon's composition is provided by the Company Commander, the team members initially learn of their capabilities via the briefing_object, for example, `:team-capability-1 (description :maneuverability 5 :firepower 6 ...)`. During the execution of the mission, if any component of a MissionAgent's platoon suffers damage, or needs to share resources, then the MissionAgent will perceive the change to its own capabilities and communicate that knowledge to the other team members, and to the supervising commanding officer via the BriefingAgent.

The plan requirements are represented in the briefing_object as the three corridors, `ohio texas utah`. There is also a `distribution` requirement that

one platoon should patrol one corridor, but leaves it up to the MissionAgents to `negotiate` their assignment to a corridor. Each corridor has its own `maneuverability :rating` requirement, which the MissionAgents match to their own `maneuverability :rating` capabilities to generate candidate roles. The MissionAgents propose, two of them negotiate a conflict, they seek and receive approval from the human commanders, and commit to their plans and begin executing the mission.

In the course of the mission, one of the platoon's advance scouts discovers an anti-tank minefield in their path. The robotic mine sweeping team is already committed to one of the other platoons, so the MissionAgent for this platoon announces his desire for any unassigned mine sweeping groups, to the whole team, while it continues to execute its role in the team plan. Shortly afterwards, the lead tank of the platoon with the mine sweepers assigned to it falls into an anti-tank trap and breaks a track. Its MissionAgent notifies the BriefingAgent and the other MissionAgents about its change in capability, and that its new subgoal is to wait for an Armored Repair Vehicle. Remembering the new plan requirement of the first MissionAgent, the mine sweeping team asks both MissionAgents (current and requesting) and the BriefingAgent, if they can be assigned to the first platoon. Both of the MissionAgents, and the BriefingAgent agree to the request, and the mine sweepers change platoons and roles.

7 Related Work

We demonstrate a proof of concept conversational case-based planning system for the team coordination of independent, intelligent software agents. From a literature review, the *SiN* [15] algorithm, which integrates a generative planner, *SHOP*, with a conversational case retriever, *NaCoDAE/HTN*, appears to be very similar to the overall MAS configuration of NaCoDAE, the BriefingAgent, and the MissionAgents in the context of a multi-agent system application. SHOP provides generalized domain knowledge that can be applied to a variety of domain problems. NaCoDAE/HTN interacts with humans to gather information specific to the domain problems. The SiN algorithm manages the matching of NaCoDAE/HTN task decompositions to SHOP plans, and the alternation of control between the two systems. The reason for integrating the two planning systems is to significantly reduce the plan space of the NaCoDAE/HTN planner, if it were used in isolation, and to allow for more interactivity with humans, if SHOP were used in isolation. Future work could investigate if the RETSINA agent HTN plan space is comparable to the NaCoDAE/HTN plan space in isolation, or to the plan space of the total SiN system.

A novel aspect of our work was to use the conversational model of NaCoDAE as a way of merging the asynchronous human- and agent-provided information into cases. To our knowledge, this is the first instance of a conversational CBR system that conversed directly with software agents. New agent queries or interactions were triggered by the regeneration of NaCoDAE's ranked list of questions in response to new input, which in turn triggered more agent responses.

Although others have used speech recognition as front-ends to conversational CBR systems [11], we did not direct any of NaCoDAE's responses back to the human user, thus NaCoDAE did not dialogue directly with the human. Rather, so as to not cognitively overload the user and to be as unobtrusive as possible, the NaCoDAE GUI was not visible, and the CCBP system passively listened to the textual transcription of human statements, attempting to extract information from the human conversation by matching their statements with the answers associated to the ranked list of questions.

8 Conclusions

The work that we have presented in this paper is significant in four ways. First, we have introduced a technique by which information technologies can be elevated to a level of accessibility that is closer to humans. Namely, by deploying NaCoDAE as a passive listener to human conversations, and by agentifying it so that it can actively dialogue with intelligent software agents in order to fill the gaps of unspecified knowledge, we showed that we can avoid overloading the human with detailed questioning while still allowing him to specify relevant information. Second, we have demonstrated the flexibility of conversational case-based reasoning at combining knowledge from many sources in a dynamic and ever-changing situation into meaningful knowledge. We do this by mapping information from human and disparate agent sources into case actions, which can then be assembled into team plans and descriptions of the environment. Also, by performing this mapping asynchronously and incrementally, NaCoDAE has demonstrated that its conversational nature is well-suited for agent information gathering domains. Third, from the perspective of team-oriented agent research, we have provided principles for developing and supporting agent teams, and tested them by applying these principles to a scenario that involved software agent teams operating in a simulated environment. Through such tests, we contribute to the understanding of agent roles and human-agent interactions in teams composed of humans and intelligent software agents. Fourth, we have provided an innovative technique to the agent community that illustrates how to access multimodal information that includes structured data as well as speech, text, and agent responses.

References

1. D. W. Aha and L. A. Breslow. Refining conversational case libraries. In *Proceedings of the Second International Conference on Case-Based Reasoning*, 1997.
2. D. W. Aha, L. A. Breslow, and T. Maney. Supporting conversational case-based reasoning in an integrated reasoning framework. *Case-Based Reasoning Integrations: Papers from the 1998 Workshop*, 1998.
3. D. Brugali and K. Sycara. Agent technology: A new frontier for the development of application frameworks? In *Object-Oriented Application Frameworks*. Wiley, 1998.
4. L. Chen and K. Sycara. Webmate: A personal agent for browsing and searching. In *Agents 1998*, May 1998.

5. P. R. Cohen and H. J. Levesque. Teamwork. *Noûs*, 25(4):487–512, 1991.
6. K. Decker, A. Pannu, K. Sycara, and M. Williamson. Designing behaviors for information agents. In *Agents 1997*, February 1997.
7. Defense Modeling, Simulation, and Tactical Technology Information Analysis Center. *ModSAF Kit C*, ModSAF 5.0 edition. http://www.modsaf.org.
8. D. J. Dwyer, J. E. Fowlkes, R. L. Oser, E. Salas, and N. E. Lane. *Team Performance Assessment and Measurement: Theory, Methods and Applications*. Erlbaum, 1997.
9. K. Erol, J. Hendler, and D. S. Nau. HTN planning: Complexity and expressivity. In *Proceedings of AAAI-94*, 1994.
10. J. A. Giampapa, M. Paolucci, and K. Sycara. Agent interoperation across multagent system boundaries. In *Agents 2000*, June 2000.
11. M. Göker and C. Thompson. Personalized conversational case-based recommendation. In *EWCBR 2000*, September 2000.
12. B. Grosz and S. Kraus. Collaborative plans for complex group action. *Artificial Intelligence*, 86(2):269–357, 1996.
13. K.-F. Lee, H.-W. Hon, and R. Reddy. An overview of the Sphinx recognition system. In *IEEE ASSP-38*, volume 1, pages 35–45, January 1990.
14. H. J. Levesque, P. R. Cohen, and J. H. T. Nunes. On acting together. Technical Note 485, AI Center, SRI International, May 1990.
15. H. Muñoz-Avila, D. W. Aha, L. A. Breslow, D. Nau, and R. Weber. Integrating conversational case retrieval with generative planning. In *Proceedings of the Fifth European Workshop on Case-Based Reasoning*.
16. M. Paolucci, O. Shehory, and K. Sycara. Interleaving planning and execution in a multiagent team planning environment. Technical Report CMU-RI-TR-00-01.
17. T. Payne, K. Sycara, M. Lewis, T. L. Lenox, and S. K. Hahn. Varying the user interaction within multi-agent systems. In *Agents 2000*, June 2000.
18. C. Prince, D. P. Baker, L. Shrestha, and E. Salas. Situation awareness in team performance. *Human Factors*, 37:123–136, 1995.
19. D. V. Pynadath, M. Tambe, N. Chauvat, and L. Cavedon. Toward team-oriented programming. In *Intelligent Agents VI: ATAL, N.R.*, pages 233–247, 1999.
20. O. Shehory and K. Sycara. The RETSINA communicator. In *Agents 2000*.
21. R. J. Stout and E. Salas. The role of planning in coordination team decision making: Implications for training. In *Human Factor and Ergonomics Society* 37^{th} *Annual Meeting*, pages 1238–1242, 1993.
22. G. Sukthankar. Team-aware multirobot strategy for cooperative path clearing. In *AAAI-2000*, July 2000.
23. K. Sycara. Multiagent systems. *AI Magazine*, 19(2):79–92, Summer 1998.
24. K. Sycara, K. Decker, A. Pannu, M. Williamson, and D. Zeng. Distributed intelligent agents. *IEEE Expert*, 11(6):36–45, 1996.
25. K. Sycara, K. Decker, and M. Williamson. Middle-agents for the internet. In *IJCAI-97*, 23–29 August 1997.
26. K. Sycara, M. Klusch, S. Widoff, and J. Lu. Dynamic service matchmaking among agents in open information environments. *Journal ACM SIGMOD Record, A. Ouksel, A. Sheth (Eds.)*, 28(1):47–53, March 1999.
27. K. Sycara and D. Zeng. Coordination of multiple intelligent software agents. *IJICIS*, 5(2 and 3):181–211, 1996.
28. M. Tambe, D. V. Pynadath, N. Chauvat, A. Das, and G. A. Kaminka. Adaptive agent architectures for heterogeneous team members. In *ICMAS-2000*, pages 301–308, 10–12 July 2000.
29. H.-C. Wong and K. Sycara. A taxonomy of middle-agents for the internet. In *ICMAS 2000*, pages 465–466, 10–12 July 2000.