

Distributed Collaboration Among Humans and Software Control Agents

C. Martin, D. Schreckenghost, P. Bonasso, D. Kortenkamp, T. Milam, and C. Thronesbery

NASA Johnson Space Center

TRAC Labs

1012 Hercules, Houston, TX, 77058 USA

1 (281) 461-9525

cmartin@traclabs.com, ghost@ieee.org

ABSTRACT

This paper describes an implemented software prototype for the Distributed Collaboration and Interaction (DCI) system, which addresses the challenges of helping humans to act as an integrated part of a multi-agent system. This area of research is unexplored to a large extent. Human interaction with agents who act autonomously most of the time, such as a process control agent in a power plant, has received little attention compared to human interaction with agents who provide a direct service to humans, such as information retrieval. This paper describes how *liaison agents* within the DCI system can support human interaction with agents that are not human-centric by design but must be supervised by or coordinated with humans. Further, the DCI prototype supports notification and planning for humans from the perspective of an organization. It treats humans in this organization as if they were agents with explicitly modeled roles and activities related to software agents in the same system. Planning for humans presents unique challenges because the models represented in traditional planners do not match well with human mental models. The DCI prototype is applied in the domain of NASA advanced life support systems, which are controlled primarily by software agents in an autonomous fashion with occasional human intervention. The DCI system provides a step toward future seamless integration of humans and software agents into a cohesive multi-agent system.

1. INTRODUCTION

As software agents become more capable and more prevalent, humans must be able to interact with a heterogeneous collection of humans and software agents. Both humans and software agents can play diverse roles in a system, with varying degrees of autonomy, initiative, and authority across different tasks. Although many of today's software agents commonly interact with humans, these interactions are often in direct service to the human in a one-on-one manner, much like a human's interaction with a travel agent, for example. However, software agents can also undertake long-term, autonomous operations that serve human purposes as well, though not *through* direct interaction with a human.

Humans need to interact with autonomous agents over long-term operation for a variety of reasons including monitoring, modifying goals, maintaining or repairing underlying hardware or software, responding to anomalies, and taking advantage of opportunities. However, research supporting human interaction with these types of agents has received relatively little attention.

As more software agents are deployed for long-term autonomous operation, the research challenge of enabling agents and humans to work together as a complete system becomes more important. To meet this challenge, research must overcome issues such as limited visibility into the agent's processing, mismatches between a human's mental models and the agent's implementation models, inadequate adjustability of the agent's autonomy, lack of notification to the human about important events at appropriate levels of abstraction, and a basic lack of compatibility between a human's interface capabilities and the interfaces provided by a software agent.

We have developed the Distributed Collaboration and Interaction (DCI) system to address these difficulties in human-agent interaction and to create an environment in which humans and software agents together can form an integrated multi-agent system. The initial motivation for the DCI system arose from our experiences with deployed intelligent control agents for NASA advanced life support systems. These intelligent control agents monitor and perform process control for regenerative life support systems, which recover usable water or air, for example, from the waste products created by biological systems over time. Over months to years of continuous operation of these agents, we discovered many unaddressed needs for human interaction. These lessons learned are detailed in [21]. Through the DCI prototype described in this paper, we have addressed and enhanced our understanding of these interaction needs and begun to formulate solutions.

Further, we have recognized the need for and provided support for interactions of groups of humans and software agents that are coordinated at the level of the organization they support. Based on our own experiences, the NASA organization (considering mission objectives and constraints) defines policies and protocols for fulfilling human roles within the organization as well as for the operation of software agents during the missions. A similar requirement to support organizational needs could be seen in the management of power plant operations, for example, if automated control agents were deployed for process control with possible monitoring and intervention by human supervisors. These types of organizations can be realized as heterogeneous human-agent multi-agent systems. In these systems, organizational policies and objectives require humans to perform in various roles and to maintain a specified level of situation awareness with respect to the automated control agents and the underlying hardware they control. In the course of their duties, humans must be able to interact with and influence the software control agents. The DCI

system supports these organizational requirements by providing an environment through which humans can act as system agents with roles and tasks managed by the organization along with software agents who perform other operations in the organization. This paper describes how the development of the DCI prototype addressed unique challenges to apply automated planning for organizational goals to human agents.

The DCI approach uses intermediate *liaison agents* associated with each human to provide an interfacing layer between the human and the rest of a multi-agent system. These liaison agents have the purpose of representing an individual human to the rest of the software in the system while supporting a human-centric and user-friendly interface for that human into the system. We also use *augmenting software* associated with other existing software in the multi-agent system (for example, a centralized planner or a particular software agent or group of these agents) to handle the human-interfacing requirements related to that domain-centric software. Although the domain-centric software must internalize some basic human-interaction functionality such as adjustable autonomy [7], our approach avoids overburdening resource-limited software agents, whose processing may include time-critical tasks, with tasks that are not directly related to their primary organizational objectives. The augmenting software is designed to integrate closely with the domain-specific software and thus off-load much of the human-centric processing.

This paper provides an overview of the DCI system and a description of a software prototype that implements many of the desired behaviors of the DCI environment. This prototype is applied to support NASA engineers working with intelligent control agents for advanced life support systems. We provide an overview of this application and a description of how the DCI prototype supports human interaction with an intelligent control agent within the context of organizationally defined roles and interaction policies. We next provide details about the implementation steps we took within the DCI environment to interface humans to a centralized planner that manages the humans' roles and activities. We then discuss some of the related research that supports this work, and we conclude with a summary and a discussion of research issues.

2. PROTOTYPE APPLICATION DOMAIN

Since 1995, we have been developing intelligent control systems for advanced life support [20; 2]. These control systems have been realized by software agents using an architecture known as 3T [1], and were designed to run autonomously for months at a time. 3T is a layered control architecture whose top tier is a hierarchical task net (HTN) planner, the plans of which are executed through a reactive middle tier that in turn manages the sensors and actuators of the hardware via a low-level control tier.

One such life support system is the advanced Water Recovery System (WRS). Developed at Johnson Space Center (JSC), the WRS is comprised of four hardware subsystems that remove the organic and inorganic materials from waste water (hand wash, shower, urine and respiration condensate) to produce potable water. From January 2001 through April 2002, the 3T system controlled the WRS autonomously in a continuous 24/7 integrated test [2]. The hardware and control software for the WRS operated unattended in JSC's Water Processing Facility (WPF) while the human hardware and software engineers performed intermittent

monitoring from remote locations (office, home, etc.) for anomalies in the system due to network, hardware, or power failures.

Over this period of operation, the humans were responsible for monitoring and occasionally intervening in WRS operations while spending the majority of their time carrying out their daily tasks on unrelated projects. Software support for this remote monitoring was minimal due to funding constraints, and remote interaction with the control system to correct problems was unavailable. Several other interaction needs were also identified in areas such as notification and improved visibility into the control system [21]. To explore support for WRS controls management, an early version of the DCI system prototype was deployed with the WRS 3T system in April 2002, and we have continued to develop the prototype and apply it to this domain through a simulation of the WRS 3T system.

Our DCI prototype supports a number of human agents, each with a set of daily activities to perform as mandated by the organization, much as NASA mission tasks would be mandated for crew members. Additionally, three of the humans are responsible for handling anomalies that occur in the WRS. One agent, the Prime, has first responsibility for responding to problems in the WRS; a second agent, the Backup, has the job of taking over if the Prime is unable to respond to the problem. The Coordinator oversees the work of the other two agents and also serves as the secondary Backup. The following section provides an overview of the DCI system and describes how the current DCI prototype supports these human agents in interacting with the WRS control agent.

3. DCI OVERVIEW AND PROTOTYPE

The philosophy behind the DCI system involves creating an environment that supports human collaboration and interaction with software agents and also giving humans appropriate tools and interfaces to participate effectively in these interactions. In short, we are striving to provide a virtual environment in which the human can interact naturally with software agents. The DCI system is made up of three types of software (1) *augmenting software* that integrates with domain-centric software or software agents to provide a human-friendly virtual multi-agent environment, (2) *liaison agents* that serve a human's interaction needs both to and from this virtual environment, and (3) *user-interface software* that is managed by the liaison agents to provide a comfortable, effective, user-friendly interface for the human to the multi-agent system.

Figure 1 depicts representative elements of a DCI system. The entities with black backgrounds (the human, the WRS system and its control agent, and the multi-agent planner) participate in, but are not part of, the DCI environment. The following paragraphs describe the function of each DCI entity in Figure 1 as well as how the entity is instantiated and used in the DCI prototype.

The Conversion Assistant for Planning (CAP) and the Event Detection Assistant (EDA) are representative pieces of *augmenting software* in the DCI system. The CAP is software that is tightly coupled and shares models with the automated planner, thereby augmenting the planner's ability to interface with human agents. Our planner is a hierarchical task net (HTN) planner known as AP that is capable of automatically monitoring and updating its plans [8]. AP evenly distributes the workload

among agents based on their capabilities and reasons about metric time for scheduling. The CAP's functionality is discussed in detail in Section 4. The EDA monitors data produced by the WRS control system and searches for patterns in this data that are of interest to the humans for such activities as anomaly analysis and performance assessment. The EDA is implemented using the Complex Event Recognition Architecture (CERA), an event detection system developed by I/Net (www.inetmi.com/whatsnew/pressrel/03-01-01.htm). As specified patterns are detected in the control data, the EDA generates and broadcasts its own events about these data patterns, which are represented at levels of abstraction suited to human understanding.

Liaison agents are central to the DCI system because they represent the human to other software agents and vice versa. The liaison agents in DCI are called Attentive Remote Interaction and Execution Liaison (ARIEL) agents, in deference to Shakespeare's Tempest character. Although the ARIEL agents in the current DCI prototype are not yet implemented as complete agents with explicit goals and desires, they do hold explicit beliefs about the state of their human user, and they exhibit many of the basic processing behaviors desired for a complete ARIEL agent implementation in the future. The beliefs held by an ARIEL agent are managed by its *State Management Service (SMS)*, which takes input from many of ARIEL's other services and creates a coherent state model of the user, including current activity, location, role, schedule, and health. The SMS also interacts with its user by querying for state input (such as schedule acknowledgements), when appropriate. Other services represented in ARIEL's design include:

- *Notification Service (NS)*: The NS combines information about the user's current state and roles, organizational policies about information distribution and situation-awareness requirements, and the user's own information preferences to determine if an incoming notice or event is of interest and, if so, how to inform the user. How to inform the user of a notice is expressed as an assessment of the "latency" and "focus of attention" [19] to be used when presenting the information, and a selection of interface modalities (e.g., computer display, pager, email) to

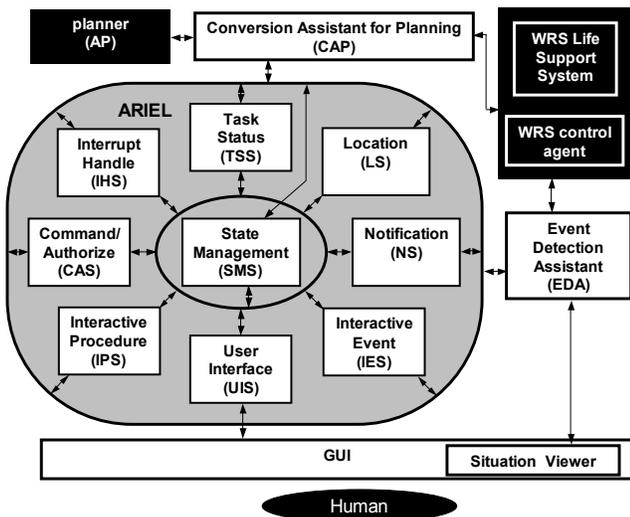


Figure 1. Representative elements of DCI architecture.

use for this presentation. The NS implementation and the representation of specifications an organization or individual may use to filter or present notices are described in detail in [19]. In the DCI prototype, the NS that is associated with the human who has the Prime role ensures that he is notified of important WRS anomaly events with high saliency. However the Backup is allowed to continue her current task without distraction because her NS simply logs anomaly notices. Notices processed by the DCI prototype include (1) events generated by the Event Detection Assistant about the WRS control system, (2) notices from the ARIEL agent to its user about ARIEL requests such as the need for schedule acknowledgement, and (3) events generated by the ARIEL agents of other humans in the user's group about various human state changes such as a location change.

- *Task Status Service (TSS)*: The TSS provides activity tracking and plan management capabilities. In the DCI prototype, the TSS (1) monitors the user for acknowledgement of time-critical assigned activities such as WRS repair tasks, (2) uses information generated by the augmented planner to inform its user when a planned task becomes ready to execute, (3) monitors its user for evidence that critical tasks have been initiated using location information or direct user queries, and (4) provides a source of feedback to the augmented planner about human progress or lack of progress toward achieving a plan. The TSS helps the ARIEL agent close the loop between the planner and a human, which is especially important for time-critical repair tasks in the DCI prototype application. The TSS implementation is discussed further in Section 4.
- *Location Service (LS)*: The LS tracks human location information including physical location and cyber location, i.e., whether or not the user is online and which display platforms she is currently using. In addition, the location service uses the combination of physical and cyber location information to infer an overall assessment of the user's *presence*, for example, "Available-Remote-Online." This location information is provided through the State Management Service to the other services and is used (1) by the Task Status Service in tracking the initiation or completion status of activities, (2) by the Notification Service in determining which notification modality is currently most appropriate, and (3) by the User Interface Service in customizing the presentation of information. In the DCI prototype, a user's physical location is tracked through log-in and log-out events within the DCI environment (via static IP mapping). We are also in the process of incorporating the use of GPS devices to track physical location. Similar versions of location tracking have been used successfully by other systems [6; 17].
- *User Interface Service (UIS)*: The UIS manages all direct interaction with the user. It invokes different modalities, such as display, pager, or email, to present information in the manner most appropriate to the user's current state and task. It also manages the overall state of information presented by any persistent user interface (e.g., a graphical user interface, GUI, versus a transient pager message) such that multiple locally-managed views of this information will remain consistent (e.g., multiple GUIs open on different display platforms). In the DCI prototype, the UIS is capable of sending emails, posting to and accepting information from the DCI user interface GUIs discussed later in this section, and accessing a simulated pager

server. Multiple interface modalities for posting information to humans have been managed successfully in related research projects such as [17]. In the future, we hope to incorporate increasingly natural and more sophisticated human-agent interaction mechanisms into this service.

The remaining ARIEL services are not yet implemented in the DCI prototype:

- *Command and Authorization Service (CAS)*: The CAS supports the user in remotely interacting with mostly autonomous agents who are tied to physical systems that the user also influences, such as the underlying WRS life support system hardware. “Commanding” refers to a user’s action of issuing directives to the underlying physical system (e.g., turning on a pump). Using DCI, this commanding should be mediated through the autonomous agent controlling the system, when possible. The CAS (1) determines if the user is authorized to command (i.e., access control), (2) ensures command lock-outs or resolves command conflicts when more than one user is interacting with the system or the control agent (e.g., the WRS control agent), and (3) reconfiguring both the automation and user interface in preparation for commanding (i.e., adjusting the autonomy of the WRS control agent).
- *Interruption Handling Service (IHS)*: The IHS coordinates the actions of other services (the Notification Service, for example) to minimize the impact of interruptions on the user’s primary tasks. Support for interruption handling includes (1) determining when the user should be interrupted and how intrusive the interruption should be, (2) mapping the human concepts of task status at interruption (delayed, deferred, suspended) to the changes needed to update the plan by an automated planner (e.g., goal changes, task completion status changes), and (3) assisting the user in managing multiple, concurrent threads of activity.
- *Interactive Procedure Service (IPS)*: The IPS assists the user in temporarily modifying standard operating procedures executed by the automated control software.
- *Interactive Event Service (IES)*: The IES assists the user in interactively defining temporary, new operational events and controlling automated monitoring for these events.

The *user interface* software in the DCI system is critical for providing humans with effective tools to participate in the overall multi-agent system. The prototype’s user interface software currently consists of GUIs that give a user access to (1) her ARIEL agent’s current model of her user state, (2) her schedule as planned and updated by the augmented planner and as annotated with status information by the Task Status Service, (3) the archive of notices that her Notification Service has determined are relevant, and (4) a high-level overview toolbar as shown in Figure 2. Additional GUIs that are in the design stages for the DCI prototype include (1) a view of the state of the user’s group, and (2) a dialogue interface that allows the user to interact conversationally with other human and software agents, including the ARIEL agents of other group members.

Figure 2 shows the primary DCI user interface toolbar. This toolbar is designed to be always visible on a small section of the user’s display and is used for multiple purposes. First, it provides buttons that allow a user to access other DCI GUIs. Second, it provides the user with “at a glance” information about important



Figure 2. DCI toolbar. Icons access additional GUIs (user state, schedule, notices, group, conversation, and logout).

changes that have recently occurred, to which the ARIEL agent is drawing the user’s attention. In Figure 2, the schedule icon has been highlighted with a red exclamation point indicating a very important scheduling event, in this case the insertion of a time-critical repair task to the user’s schedule as the next task the user needs to perform. (This icon actually flashes on and off and makes a sound in the prototype to better draw the user’s attention.) The notice icon in Figure 2 also draws the user’s attention to a lesser extent by indicating that two highly important or urgent notices are available. The toolbar shown in Figure 2 can be used to quickly orient the user to the current operational situation when she logs in to the DCI system and to alert her to what has transpired while she was offline. It also provides situation awareness as she works at other tasks while logged in to the system. The toolbar implementation uses saliency annotations from the *Notification Service* and the *Task Status Service* to determine whether to concentrate the user’s focus of attention (with flashing icons and sounds) or to rely on the user’s peripheral awareness to detect changes (with simple icon changes as in [5]).

The *Situation Viewer* provides one further element needed in an effective user interface for humans interacting with intelligent control agents. In the DCI prototype, the Situation Viewer summarizes complex situations recognized by the Event Detection Assistant and provides needed visibility into the operation of the WRS control system. Previous work in viewing situations focused on providing organized logs [24]. This work integrates discrete and analog events at multiple levels of abstraction provided as part of the situation data structure.

The DCI prototype is implemented as a distributed system of approximately 30 processes using both CORBA and IPC [22] for distributed communications. The user interface software and most of the ARIEL services are implemented in Java and have been executed at various times on Linux, Windows, and Macintosh systems. The CAP augmentation for the planner, each Task Status Service, and the Event Detection Assistant using CERA are implemented in Lisp and execute on Linux systems.

This overview of the DCI prototype shows how the DCI system supports human interaction with the WRS intelligent control system. Although we limit our in-depth discussion in the following section to applying automated planning for humans, similar design and implementation challenges arose for each piece of the prototype.

4. PLANNING FOR HUMAN AGENTS

The DCI prototype explores applying an automated planner to plan for humans from the perspective of the NASA organization, for both daily tasks as well as for unexpected tasks due to anomalies that arise from the WRS operations. This section describes (1) the overall challenges we have observed for applying automated planning to human agents, (2) the approach

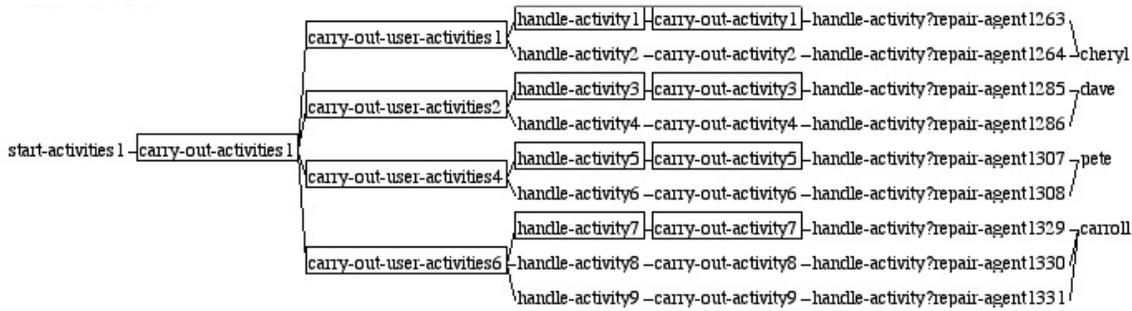


Figure 3. A typical group plan generated for four agents by the automated planner AP [8]. The hierarchy is read from left (most abstract) to right (plan leaves). Highlighted tasks are ready to be executed.

taken in the DCI prototype to explore possible solutions to these challenges, and (3) the resulting support provided by the DCI prototype for the WRS controls domain to which we have applied the prototype. Our initial exploration of these challenges in the DCI prototype provides a step toward future, more general solutions across application domains.

Applying an automated planner to human agents poses interesting research issues because autonomous planners do not integrate with humans in the same manner as they do with other software agents, such as layered control architectures like 3T [1]. These issues arise from the different ways in which automated agents and humans treat the plan. An agent like 3T will perform all tasks in the plan it receives in a pre-specified manner. On the other-hand, a human agent interprets a plan generated for him or her at a higher level of abstraction. Even if he or she intends to follow the plan there exists less inherent predictability about how and when the tasks will be accomplished. In addition, humans and software agents interact with planners differently. Software agents can be very responsive, even for low-level operations. For example, the WRS control agent will acknowledge every directive and execute plan steps as soon as possible. In contrast, humans are less responsive and would find frequent interaction with the planner burdensome. For example, humans may fail to acknowledge tasks before starting to execute them or fail to provide evidence that tasks have been completed. Further, software agents can easily understand a planner's representation of a plan through common semantics and data structures. However, a human, concerned primarily with what she must do and when, needs a different representation of the generated plan. For example, compare the view of a plan generated by AP in

Figure 4. DCI personalized schedule view. Plans from the automated planner appear to the user in a natural form. Tool tips over the activity names show the status of a given activity currently modeled by the user's ARIEL agent.

Figure 3, to the personalized daily schedule view of a similar plan provided through the DCI prototype for a given user in Figure 4.

4.1 System Design

The DCI system accommodates the differences between planning for software agents and planning for human agents by adding software to mediate between the human and the automated planner perspectives. By encapsulating the capabilities supporting each perspective in its own process, we handle separately (1) integration with the planner perspective and (2) integration with the human perspective. The interaction among DCI components providing this mediation for planning is shown in Figure 5. Note that multiple humans would correspond to multiple ARIEL agents in this figure, all of whom interact with a single planner and its associated augmentation software.

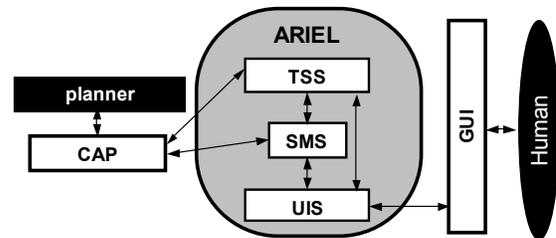


Figure 5. Information flow among DCI components mediating between a human and an automated planner

First, DCI handles integration with the planner perspective through the Conversion Assistant for Planning (CAP). The CAP conditions plans generated by the planner for use by humans and their ARIEL agents, and it converts information coming from humans and their ARIEL agents to the planner's perspective. Beyond these interpretation duties, the CAP is responsible for monitoring the execution of the group plan and to initiate updating the plan (e.g., prompting the planner to replan) when the domain situation changes or when agents become unavailable.

Second, DCI handles integration with the human perspective through the ARIEL agent and corresponding user interfaces. ARIEL's Task Status Service (TSS) is responsible to track human activities and to model when new activities are ready to execute and when ongoing activities are completed or overcome by events. The State Management Service (SMS) creates a user-centric model of the plan provided by CAP and annotates this model with status information from TSS. The User Interface Service (UIS) uses the SMS model of the plan to generate a

schedule model that is the basis for the schedule view seen by the user. The ARIEL agent serves both the human, by providing a useable schedule interface (as in Figure 4), and the planner, by providing feedback about task completion and acknowledgment, which humans do not typically provide in a manner that is easily used for computation.

4.2 Prototype Application

The remainder of this section describes how we used this approach to support centralized planning for human agents in the DCI prototype applied to the WRS domain. The planner controls task assignment, but does not dictate the low-level details of how the humans carry out their tasks. Note that this type of centralized control for human activities is needed to achieve the organizational goals of humans such as NASA crew members on a mission, during which their time serves as a critical resource and their activities are highly regimented. The following typical WRS scenario is provided for context:

Initially, the planner generates a plan of daily tasks for the human agents such as the plan shown in Figure 3. Once the initial plan is in effect, the CAP monitors the status of the WRS for anomalies that require human intervention. When the CAP receives evidence of such an anomaly, it invokes the planner to generate a new group plan that encompasses a repair task for the anomaly, using the agent roles and responsibilities previously described. The new plan usually means a schedule alteration for the human who is on-call for the WRS (the Prime). The CAP and the Prime's ARIEL agent work together to inform the Prime that her current task must be interrupted and a new time-critical repair task must be performed. The Prime's ARIEL agent monitors the user's display interface and the user's location for evidence that she has accepted and initiated the new task and provides feedback to the CAP about the status of this activity. If the Prime is not able to acknowledge or initiate the repair task within an appropriate time, the CAP interprets the resulting status events generated by her ARIEL agent as a temporary unavailability of this human agent for time-critical tasks, and initiates replanning with this information. AP will reassign the task to the Backup agent, and the CAP will use the resulting plan and knowledge of the state of execution of the other daily tasks to reassign the remaining daily tasks to the Prime agent. Although this scenario seems straightforward, each DCI component shown in Figure 5 performs critical processing to mediate this interaction.

In the DCI prototype, the CAP (1) recognizes when situations occur requiring an update of the group plan including both the initial need for a repair task and any agent availability problems that may occur, (2) recasts planning information provided by the planner for the user, and (3) maintains continuity of activities across plan updates to support a human's perception of activities. For example, although the concept of "interruption" is a natural way for humans to mentally model task status, AP, like most AI planners, does not model the notion of a task placed in abeyance until another more critical task is executed, such as the WRS repair task. In AP the successful execution of a task should bring about a set of desired effects in AP's perceived situation. These include effects from sensed data, like a communication from 3T that a problem has been corrected, as well as effects added by AP when a task is completed such as the purpose clause (a first order proposition) of a given plan operator. If these effects are not in the perceived situation when a new plan is generated, and the task

purpose is still a desired goal for the user, the task will again appear in the new plan. The CAP needs to properly interpret this reappearance for the user to avoid redundant updates, and to correctly indicate whether a task has indeed been interrupted. By comparing the task purpose with that of previously assigned tasks, checking for the existence of the effects of the task in the perceived situation, and keeping track of the initiation status of events, the CAP can determine whether a given task has been interrupted.

The overall functions of the TSS with respect to activity tracking and plan management are described in Section 3. At a finer level of detail, the TSS (1) models and reasons about task criticality to determine whether to request user acknowledgement and when to focus the user's attention on a particular activity and (2) assesses changes in activity status using both direct activity feedback from the user and indirect evidence. With respect to the first point, the TSS distinguishes time-critical tasks such as WRS repair tasks from other, non-critical tasks such as routine daily activities. The TSS requires more responsiveness and interaction from the user for time critical tasks.

To assess activity status, the TSS may make assumptions about the status of non-critical tasks, but requires evidence to infer status changes for time-critical WRS repair tasks. When a time-critical task is assigned to a user, his TSS begins watching for an acknowledgement that he has accepted the task. As the current time approaches the timeout limit for acknowledgement of the task, the TSS attaches an increased saliency to the task, which causes the User Interface Service to attempt to focus his attention on the request for acknowledgement. If he does acknowledge the assignment of the time-critical task within the allotted time period, the TSS begins monitoring for indirect evidence that the user has initiated the task, such as the arrival of the agent at the location where the activity should take place, e.g., the Water Processing Facility. The TSS also monitors for direct evidence that the WRS repair task has been completed, e.g., in the case of lost communications within the WRS system, the evidence comes in the form of a message from the control software agent that the communications have been restored. The TSS provides all status updates to the CAP, allowing the CAP to effectively manage coordination of the human agents with respect to the WRS and find a new agent to perform a given repair task, if necessary.

For non-critical daily tasks, the current TSS implementation in the DCI prototype minimizes user distraction by making assumptions about task initiation and completion. The TSS assumes these tasks are initiated and completed at the times they are scheduled, as long as the human has previously acknowledged her acceptance of the most recent overall daily schedule. Future TSS implementations will involve more instrumentation of the user for these routine tasks and will allow after-action task-status reconciliation with the user to confirm, for example, whether tasks inferred complete were actually performed.

The discussion in this section clearly shows that the interleaving of time-critical tasks with daily human activities through automated planning requires special considerations, and the CAP and the ARIEL agents must work together to keep the human and planner up to date with respect to scheduled and executing tasks. Through this coordinated, intelligent interplay between the CAP and the ARIEL agents, the DCI architecture compliments and extends the capabilities of the automated planning algorithms so

that they can better serve the human users. As shown in the DCI overview in Section 3, these capabilities are an important piece of the overall objective to support human interaction within a multi-agent system.

5. RELATED WORK

To integrate humans into a multi-agent system alongside software agents, we have leveraged existing research across a wide range of areas including, human-agent interaction [15; 12], teaming and human-agent teams [6; 14; 4], user interfaces and underlying applications [17; 5], characteristics of autonomous agents including adjustable autonomy [7; 16], and planning tools and mixed-initiative planning [9; 13; 11]. This section highlights some of this research and how it applies to DCI.

A very successful and innovative implementation of interaction between humans and software agents has been demonstrated in the Electric Elves system to support human organizations [6]. This system incorporates multiple humans and multiple software agents; however, each human interacts primarily with the capabilities of his or her own “proxy” (or with non-autonomous software accessed through the proxy). The Electric Elves architecture does not fully address our requirements for support agents to act as mediators and/or enablers for humans to interact with yet a third class of agents: autonomous control systems.

The interface agents in the MokSAF environment support human interaction with software agents in human/agent teams in the domain of decision-support for military route planning [14]. These interface agents assist humans in tasking other agents (route-planning agents), present situation information to the human team members, and help humans communicate and coordinate with other humans. All of these capabilities are desirable for ARIEL agents in DCI, and the MokSAF work identifies many important issues with respect to enabling an interface agent to act on a user’s behalf. However, the MokSAF environment does not address DCI’s need to interact with mostly autonomous agents because the route-planning agents are still human-centric in that their primary purpose is to assist a human in generating a route, if tasked by the human to do so.

Bradshaw et al have investigated human-agent teamwork in depth [4]. For example, they are modeling human-robot collaboration, using Brahms to simulate human work practice [4]. They continue to develop policies to support agent interaction and teamwork, based on KAoS agent services [3]. This work represents the type of basic theories and types of services to enforce interaction policies that are needed to support further DCI implementation of increasingly complex human-agent interaction.

Few research efforts in AI planning are focused on interpreting the output of an automated planner for a human agent to whom the plan applies. There have been many efforts to make it easier for a human planner to use automated planning tools [13; 11] and mixed-initiative planning [9]. Also, several planners, like AP [8], have been designed to manage the activities of large groups of humans such as military units, but not individuals [25; 23]. The output of NASA’s automated aids for planning the daily activities of shuttle and station crew is either interpreted by another human (e.g., the CapCom) or transformed into paper schedules managed by humans [10]. In contrast, the DCI system needs a planner that not only generates and updates plans for individual humans, but

also interprets the plans so that the humans have a ready understanding of their current and future tasks [18].

6. CONCLUSIONS

This paper describes the Distributed Collaboration and Interaction (DCI) system design and an implemented prototype of that design. DCI aids human interaction with complex, mostly autonomous, domain-centric software agents in the context of an integrated multi-agent system supporting organizational goals and policies. In addition to an overview of DCI, we provide a detailed description of the challenges we faced to apply multi-agent planning to human agents. Through the DCI environment, humans can become part of the multi-agent world and act naturally within it. As autonomous software agents become more common, both in environments that highly regiment the activity of human agents such as NASA mission support and in every-day environments such as smart houses, supporting a human’s ability to interact with these software agents becomes increasingly important.

In the development of the DCI prototype, we have addressed many interaction challenges including (1) mapping models and data designed for use by software (and thus containing artifacts of implementation choices) to human-usable information, (2) balancing an organization’s need for human awareness of software agent activities with the need to avoid both information and cognitive overload, and (3) providing necessary feedback about human state to automated software that is difficult or annoying for humans to provide manually. These interactions are directed both toward the human and toward other software agents. Our next step is to provide fully interactive multi-step interactions between humans and autonomous control agents.

We have seen the need for integration of humans with mostly autonomous software agents through our experiences with autonomous control agents for NASA life support systems. Supporting this type of multi-agent system, including planning for humans as agents, is a relatively novel endeavor. Through our work to prototype the DCI system, we hope to discover a set of design principles for building systems like this in the future. Thus far, we have seen that:

- Neither thin “wrappers” around software agents nor sophisticated user interfaces are enough to support human interaction with complex software agents. Our experience indicates that active and vigilant processing, based on knowledge of the complexity of the software agents as well as knowledge of the needs of the human, is required to allow the human to manifest herself effectively into a multi-agent world. The concept of a liaison agent fits this role well.
- For effective interaction, complex software agents must implement some human-centric functionality such as the capability for adjustable autonomy allowing humans to supervise and adjust the agents’ behavior if necessary. However, much of the human-centric processing can be handled by tightly coupled external software to avoid overloading the software agents and degrading their performance on their primary task.

Based on our experiences, human interaction with complex software agents (who themselves interact with one another and who have their own, independent goals) has received relatively little previous attention. We have designed the DCI system to

address these issues and applied a DCI prototype in a NASA-oriented domain. This DCI prototype represents an important step toward integrating humans with multi-agent systems in the future.

7. ACKNOWLEDGEMENTS

We want to acknowledge the support of Dr Michael Shafto, the manager of the Human-Centered Computing topic in NASA's Intelligent Systems Program, under which this work was done.

8. REFERENCES

- [1] Bonasso, R. P., Firby, J. R., Gat, E., Kortenkamp, D., Miller, D. P., and Slack, M. G. Experiences with an Architecture for Intelligent, Reactive Agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 9 (1997). 237-256.
- [2] Bonasso, R. P., Kortenkamp, D., and Thronesbery, C. Intelligent Control of A Water Recovery System: Three years in the Trenches. *AI Magazine* 23 (4). (2002).
- [3] Bradshaw, J. M., Duffield, S., Benoit, P., and Woolley, J. D. KAoS: Toward an Industrial-strength Generic Agent Architecture. In *Software Agents*, Bradshaw, J. M., ed. AAAI Press/ The MIT Press, Cambridge, MA, 1997. 375-418.
- [4] Bradshaw, J. M., Sierhuis, M., Acquisti, A., Feltovich, P., Hoffman, R., Jeffers, R., Prescott, D., Suri, N., Uszok, A., and van Hoof, R. Adjustable Autonomy and Human-Agent Teamwork in Practice: An Interim Report on Space Applications. In *Agent Autonomy*, Hexmoor, H., Falcone, R., and Castelfranchi, C., eds. Kluwer, To Appear.
- [5] Cadiz, J., Venolia, G. D., Jancke, G., and Gupta, A. *Sideshow: Providing Peripheral Awareness of Important Information*, Technical Report. MSR-TR-2001-83, Microsoft Research, Redmond, WA, 2001.
- [6] Chalupsky, H., Gil, Y., Knoblock, C. A., Lerman, K., Oh, J., Pynadath, D. V., Russ, T. A., and Tambe, M. Electric Elves: Applying Agent Technology to Support Human Organizations. In *Proceedings of Innovative Applications of Artificial Intelligence* (Seattle, WA, 2001).
- [7] Dorais, G. A., Bonasso, R. P., Kortenkamp, D., Pell, B., and Schreckenghost, D. Adjustable Autonomy for Human-Centered Autonomous Systems on Mars. In *Proceedings of Mars Society Conference*, 1998).
- [8] Elsaesser, C. and Sanborn, J. An Architecture for Adversarial Planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 20, 1 (1990). 186-194.
- [9] Ferguson, G. and Allen, J. F. TRIPS: An Integrated Intelligent Problem-Solving Assistant. In *Proceedings of Fifteenth National Conference on Artificial Intelligence (AAAI-98)* (Madison, WI, 1998).
- [10] JSC. *Consolidated Planning Systems (CPS) Users Manual*, Johnson Space Center, Houston, TX, 1999.
- [11] Knoblock, C. A., Minton, S., Ambite, J. L., Muslea, M., Oh, J., and Frank, M. Mixed Initiative Multi-source Information Assistants. In *Proceedings of Proceedings of the 10th International World Wide Web Conference* (Hong Kong, 2001). www10.org/cdrom/papers/frame.html.
- [12] Lewis, M. Designing for Human-Agent Interaction. *AI Magazine* 19 (2): 67-78. (1998).
- [13] Myers, K. L., Tyson, M. W., Wolverson, M. J., Jarvis, P. A., Lee, T. J., and desJardins, M. PASSAT: A User-Centric Planning Framework. In *Proceedings of 3rd International NASA Workshop on Planning and Scheduling for Space* (Houston, TX, 2002). Institute for Advanced Interdisciplinary Research.
- [14] Payne, T. R., Sycara, K., and Lewis, M. Varying the User Interaction within Multi-Agent Systems. In *Proceedings of Fourth International Conference on Autonomous Agents* (Barcelona, Catalonia, Spain, 2000). ACM Press. 412-418.
- [15] Rich, C. and Sidner, C. L. COLLAGEN: A Collaboration Manager for Software Interface Agents. *User Modeling and User-Adapted Interaction*, 8, 3-4 (1998). 315-350.
- [16] Scerri, P., Pynadath, D. V., and Tambe, M. Adjustable Autonomy in Real-World Multi-Agent Environments. In *Proceedings of Autonomous Agents* (Montreal, Canada, 2001). ACM Press. 300-307.
- [17] Schmandt, C., Marmasse, N., Marti, S., Sawhney, N., and Wheeler, S. Everywhere Messaging. *IBM Systems Journal*, 39, 3&4 (2000). 660-677.
- [18] Schreckenghost, D. and Hudson, M. B. Automation in Context: Planning Manned Space Exploration Activities. In *Proceedings of ISAIRAS* (Montreal, Canada, 2001).
- [19] Schreckenghost, D., Martin, C. E., and Thronesbery, C. Specifying Organizational Policies and Individual Preferences for Human-Software Interaction. In *Proceedings of AAAI Fall Symposium on Etiquette for Human-Computer Work* (North Falmouth, MA, 2002).
- [20] Schreckenghost, D., Ryan, D., Thronesbery, C., Bonasso, R. P., and Poirot, D. Intelligent Control of Life Support Systems for Space Habitats. In *Proceedings of Tenth Conference on Innovative Applications of Artificial Intelligence* (Madison, WI, 1998). AAAI Press / The MIT Press. 1140-1145.
- [21] Schreckenghost, D., Thronesbery, C., Bonasso, R. P., Kortenkamp, D., and Martin, C. E. Intelligent Control of Life Support for Space Missions. *IEEE Intelligent Systems* September/October: 24-31. (2002).
- [22] Simmons, R. and Dale, J. *Inter-Process Communication: A Reference Manual. IPC Version 6.0*. CMU Robotics Institute, 1997.
- [23] Tate, A., Dalton, J., and Levine, J. O-Plan: A Web-based AI Planning Agent. In *Proceedings of Seventeenth National Conference on Artificial Intelligence* (Austin, TX, 2000). AAAI Press. 1131-1132.
- [24] Thronesbery, C., Christoffersen, K., and Malin, J. Situation-Oriented Displays of Shuttle Data. In *Proceedings of Human Factors and Ergonomics Society 43rd Annual Meeting* (Houston, TX, 1999).
- [25] Wilkins, D. and Myers, K. A Multiagent Planning Architecture. In *Proceedings of Artificial Intelligence Planning Systems* (Pittsburg, PA, 1998). 154-162.