# INTELLIGENT AGENTS FOR SCHEDULING SPACE COMMUNICATIONS

**Pete Bonasso, Debra Schreckenghost**
*TRACLabs, Inc., 1012 Hercules, Houston,. TX 77058*
bonasso@traclabs.com, schreck@traclabs.com

## ABSTRACT

The evolving Deep Space Network (DSN) scheduling architecture will need a radically new user interface paradigm that must allow DSN missions to both unequivocally specify their requests and integrate them with those of other users in increasingly crowded bandwidths. We have designed and prototyped mission software agents that interface with the existing DSN scheduling engine (DSE). These agents use models of mission preferences for scheduling requests, conflict resolution and notifications, and take actions on the part of the user to resolve schedule conflicts or take advantage of unexpected asset availability. This paper describes the design an initial prototype of that system.

## 1. MOTIVATIONS

Recent planning for human exploration to the Moon and beyond as well as maintaining vibrant space and Earth science programs resulted in a new concept for the communications architecture. Though an important hallmark of the future architecture will be advanced resource optimization software to manage the oversubscribed communications assets, equally important will be a radically new user interface paradigm to that software that must allow space communications missions to both unequivocally specify their requests and also iteratively get those requests integrated with those of other users in increasingly crowded bandwidths. It is this last aspect of the interfaces – ability to interactively collaborate with other users and the resource scheduling software -- that will ensure a successful communications support architecture. What makes the development of such an interface a significant challenge is that in the new regime, mission operations staff (hereafter called user scheduling representatives, or user reps) will be given direct control of the schedules of communications assets, allowing them to directly change the request (e.g., antenna resources and timeframes) while working with other user reps to solve scheduling conflicts in a collegial environment [1]. Such an interface must be able to:

1) Assist user reps in generating clearly specified communications requests and tracking their status before, during and after each mission
2) Interface with resource scheduling engines and private workspaces of schedules and asset configurations so the user rep may examine alternative requests in what-if scenarios
3) Take action on the part of the user rep for routine schedule management as allowed by the mission preferences
4) Intelligently support peer-to-peer interaction with other user reps to resolve scheduling conflicts

We believe such an interface cannot be developed easily with conventional means, but instead is best designed using intelligent agent technologies, resulting in an intelligent space communications scheduling agent for each user scheduling representative. Such an intelligent agency is the Distributed Collaboration and Interaction (DCI) system [2], developed by TRACLabs for other NASA projects, which employs liaison agents for each user, designed to interact with both other liaison agents as well as with intelligent software such as automated planners and schedulers. Therefore, to meet the scheduling needs described above we:

1. Designed and developed DCI scheduling agents to interface with existing space communications scheduling engines using a local working database of active schedule possibilities
2. Extended the existing DCI capabilities to model user preferences for communications requests, conflict resolution and notification of schedule changes
3. Allowed the user reps to vary the autonomy of the scheduling agent
4. Extended the existing DCI agency to accommodate planful interactions for peer-to-peer resolution of schedule conflicts

In this project we worked with Deep Space Network (DSN) user schedule representatives to identify benchmark scenarios and use cases, and from those scenarios and from knowledge of the evolving design of the DSN Service Scheduling Software ($S^3$) architecture we derived command and information requirements to support user scheduling reps in their schedule management activities. From these efforts we determined that the DCI system could support the development of resource requests, the interaction with a scheduling engine for investigating mission alternatives, and the solving of schedule conflicts, first by having the

*Proc. 'IJCAI–09 Workshop on Artificial Intelligence in Space', Pasadena, California, US*
*17–18 July 2009 (ESA SP-673, September 2009)*

requests modified based on user rep preferences and later, by peer-to-peer conflict resolution techniques.

This paper describes the design of DCI for DSN scheduling that meets the requirements of the developing $S^3$ architecture as exemplified in the scenarios and use cases. Further, we describe a software demonstration that showed DCI supporting user reps by monitoring the master schedule for asset and schedule changes, responding to these changes by automatically taking action on the part of the user rep, based on user-allowable actions, and providing a facility for the user reps to launch a pathfinder scheduling engine graphical user interface (GUI) for $S^3$ [3] to display the results of the actions.

1) A single user scenario wherein a user rep prepares a request for execution. Use cases are:
   a) User rep prepares a request and the supporting DCI agent runs the auto-repair functions for resolving conflicts and for resolving violations
   b) After the master schedule is run, previously published requests are violated; the $S^3$ software sends the violated requests to the DCI agent, which automatically suggests and runs request changes to produce a problem-free schedule
   c) An asset becomes available and the $S^3$ software sends the new information to the DCI Agent, which automatically revokes relaxations on previous requests, successfully schedules them into and presents the results to the user rep

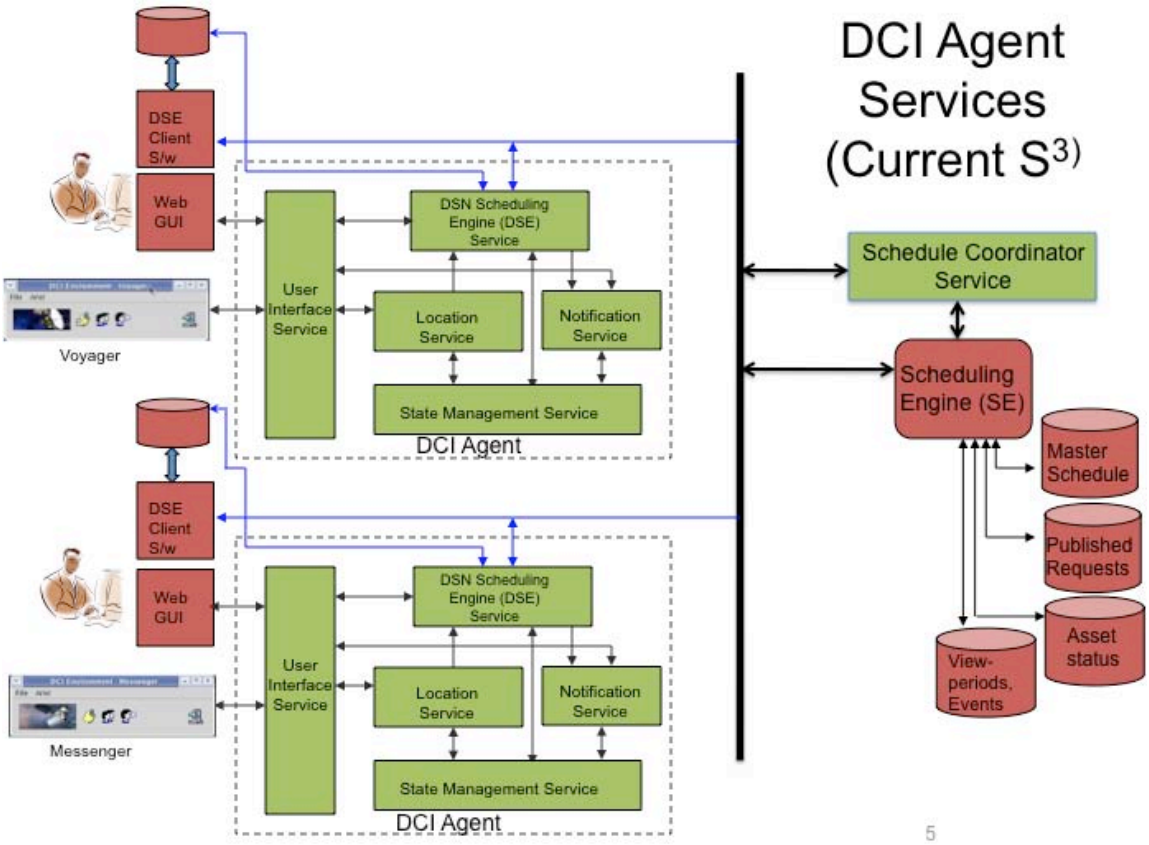


Figure 1 Detailed view of DCI in the S3 framework. In addition to the services described in the text, the displays from both DCI, such as the notice viewer at the bottom of the picture, and the DSE-client schedule

## 2. SCENARIO DEVELOPMENT

The scheduling scenarios take place at least eight weeks before the first mission sequencing (uploading command sequences to the spacecraft). This allowed us to investigate more user options, since those options will be more limited once the sequencing has begun. The scenarios and use cases to investigate how the DCI services can benefit the user reps are:

2) A multi-user scenario, wherein the master schedule is run and violations occur that need at least two user reps to modify their requests for a resolution: the DCI agents of the affected user reps establish information exchange channels to allow the affected user reps to view each other's request and a common presentation of the relevant portion of the affected schedule

## 3. CONTROL & INFORMATION (C&I) REQUIREMENTS AND DESIGNING THE DCI FRAMEWORK

The current Service Scheduling Software ($S^3$) architecture and DCI's integration with it is shown in Figure 1. On one side of a firewall is the service scheduling software application. $S^3$ services are made available to DSN user reps through a web interface, including access to the schedule and asset databases. User reps use private workspaces to develop requests and test them with the scheduling software before publishing them to the master schedule.

With this design, instead of having to help another group of schedulers understand their requests and then check the resulting schedule, the user reps will be able to directly change the requests in the schedule for their mission. Also available will be various web services supporting group collaboration, with such technologies as instant messaging and chat. And of course the user reps are linked via email and phone as well.

Each user rep will have a software agent running locally 24/7 and integrated with the $S^3$ applications. The agent would manage notifications and the exercise of the scheduling software. These mission agents would be designed from our DCI system. Additionally we see a need for a Schedule Coordinator Service to monitor for and report schedule changes and track and asset changes, and to serve as a timekeeper for user rep negotiations to solve critical schedule problems.

Each DCI agent is essentially a collection of services tailored for the supported user working in the target application domain. Figure **1** shows our selection of DCI services and the flow of information and commands derived from the scenario use cases.

There will be one agent for each user scheduling representative. On each user rep's display there is a tool bar from which one launches the graphical interfaces supporting various actions the user rep might take. Underneath the hood is the suite of services provided by each DCI agent. DCI can tailor the number and types of services for different applications. For the DSN application we've selected services for providing a user interface to the agent, user location tracking, notification, user state management and a DSE service that mediates user interaction with the DSN scheduling engine.

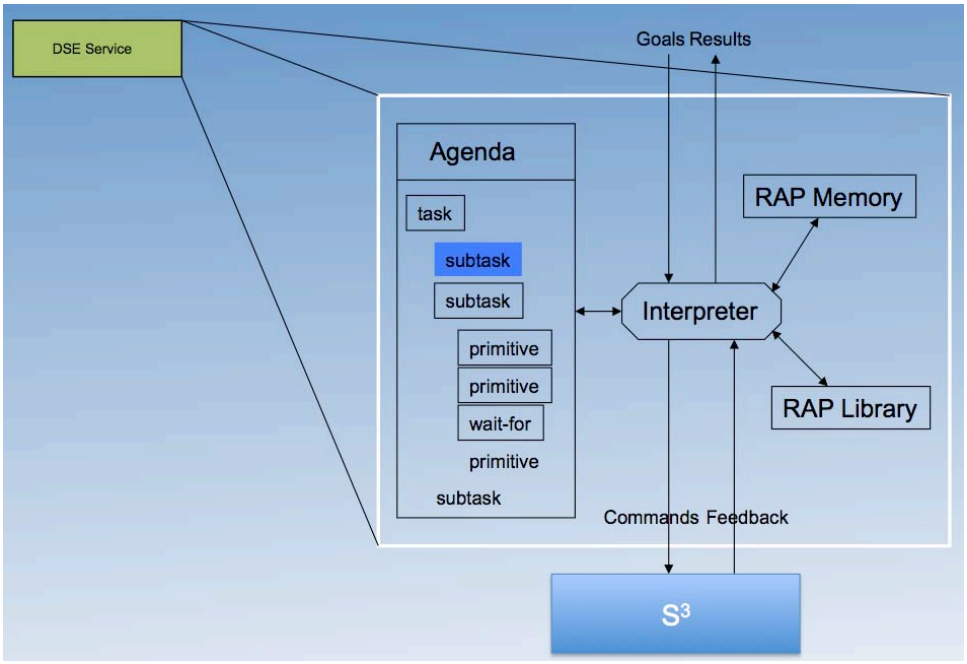


Figure 2 The DSN Scheduling Engine (DSE) Service.

The state management service (SMS) uses a blackboard memory model to maintain a consistent picture of the user state, and to maintain information on mission requests and service configurations, as well as the list of user-allowed actions that can be taken by the DSE service.

The location service (LS) keeps track of the online status and availability of the user rep and her backup.

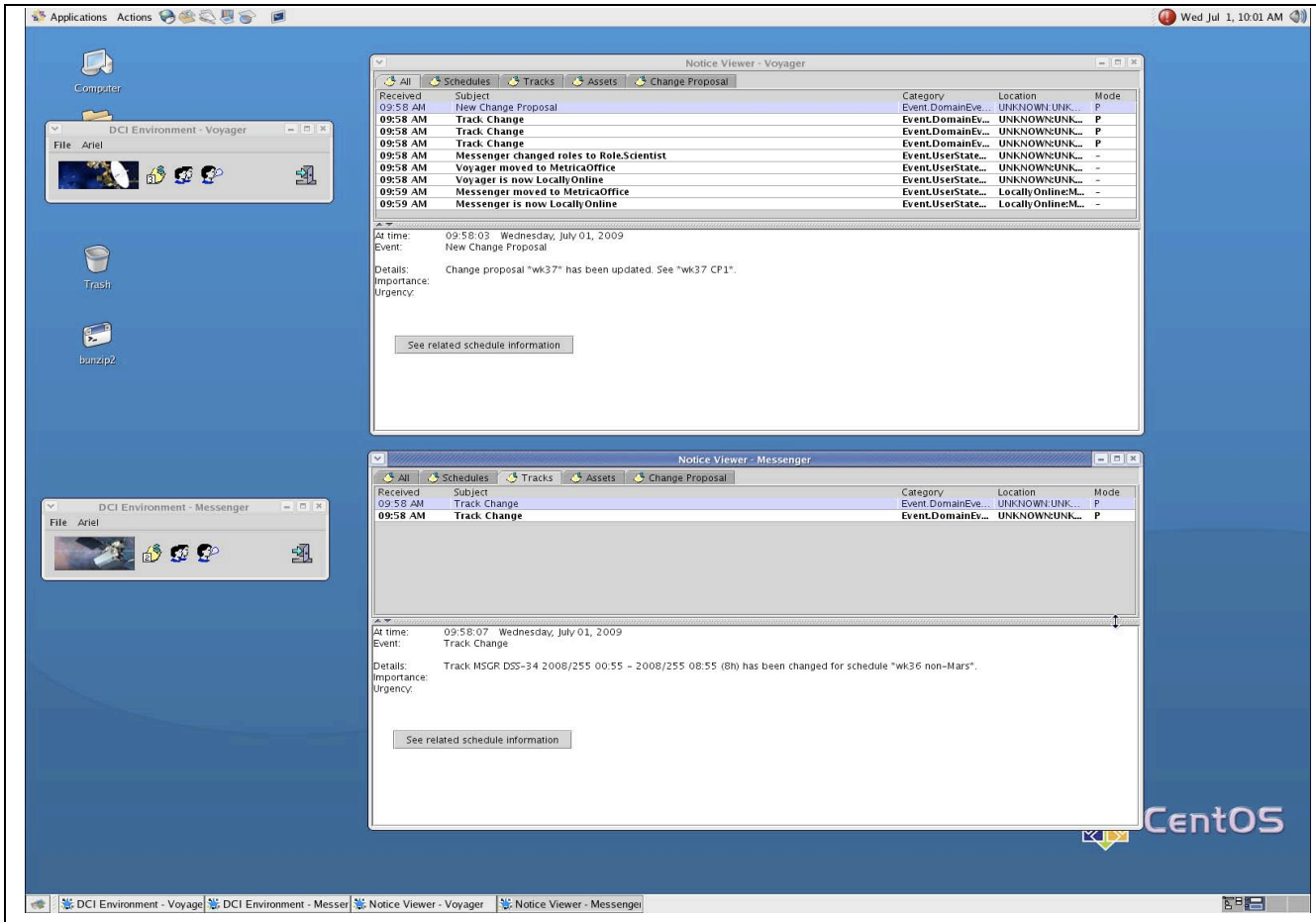


Figure 3 Toolbars and Notice viewers for the Voyager and Messenger missions. Each would actually be running on the desktop of the individual schedule managers, but we have consolidated them here for the demonstration. At the beginning of the scenario, both agents have previous messages that have yet to be reviewed, covering such things as the online status of other schedule managers, previous track changes and coordination information for change proposals. Some of the messages also have launch buttons for bringing up the DSE-client to view information related to the notice. The tabs at the top of the notice viewer group the notices according to category or discussion threads.

The notification service (NS) uses pattern matching to filter and annotate incoming notices by comparing a set of rules associated with the user's role to incoming notices. The annotation identifies how much latency can be tolerated in notifying the user and whether the user's attention should be shifted to the notice.

The user interface service (UIS) manages the presentation of information obtained from the other services. It provides user interaction with notices, including launching context sensitive displays of data associated with a notice, such as the schedule resulting from a new request. It can also support paging, email and viewing information common to two or more users, such as schedules resulting from multi-user negotiations. The UIS maintains a persistent model of the information obtained from the other services that is independent from the manner of displaying that information. This allows us to easily accommodate web-based displays of information from the agent in the $S^3$ framework.

The DSE service is the main innovation we've developed for the DSN application (Figure **2**). It executes actions on the part of the user and interacts directly with the $S^3$ scheduling engine (SE). This service uses a reactive planning engine to take planful actions on the part of the user, based on user-allowed actions maintained in the SMS. These reactive plans are stored and managed by the RAPs system that TRACLabs has used over the years in support of robotic and process control applications [4]. Plans include those for managing requests, repairing schedule problems, monitoring and responding on the part of the user rep to asset and track changes, and maintaining the status of a given user's participation in multi-user conflict negotiations.

The DSE service also monitors communications from the schedule coordination service (SCS) and can send commands and receive information from the S[3] software.

update, the change significantly affected a Voyager request. So that agent responded by first attempting to resolve the scheduling problems, and failing that, replaced a previous request with a fallback for that schedule week.

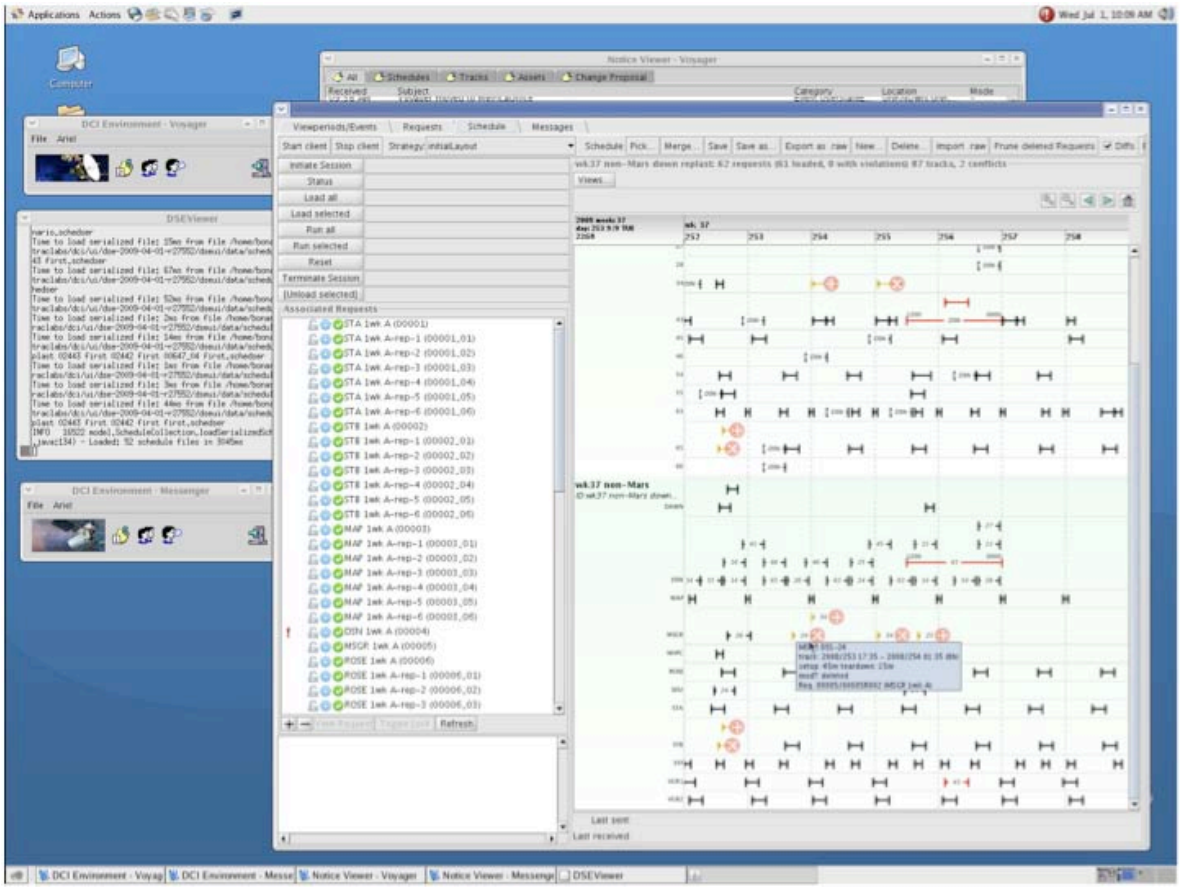

Figure 4 Launching of the DSE-client to Show Messenger's Track Changes.

All of the DSE service plans are variably autonomous. That is, they can be tagged with annotations specified by the user as to whether to run automatically or under user control. Variable autonomy permits the user to gradually automate scheduling functions performed by the agent as he or she gain insight and trust in agent behaviours with use.

## 4. DEMONSTRATION

We developed a first-order software prototype of the above-described DCI system, involving two agents, one for the Voyager 1 mission and one for the Mercury mission, Messenger, situated in a mid-term planning horizon. There were two main parts to the demonstration. In the first, the schedule coordinator service (SCS) broadcast a change in the availability of a 70-meter antenna. While both agents received the

In the second part of the demonstration, the SCS broadcast a message that some of the time on the same antenna is restored, and the Voyager agent successfully restored the original request to the schedule. Details of the demonstration follow.

### 4.1 Part One

A screenshot of the agent toolbars and their notice viewers at the beginning of the demonstration is shown in Figure **3**.

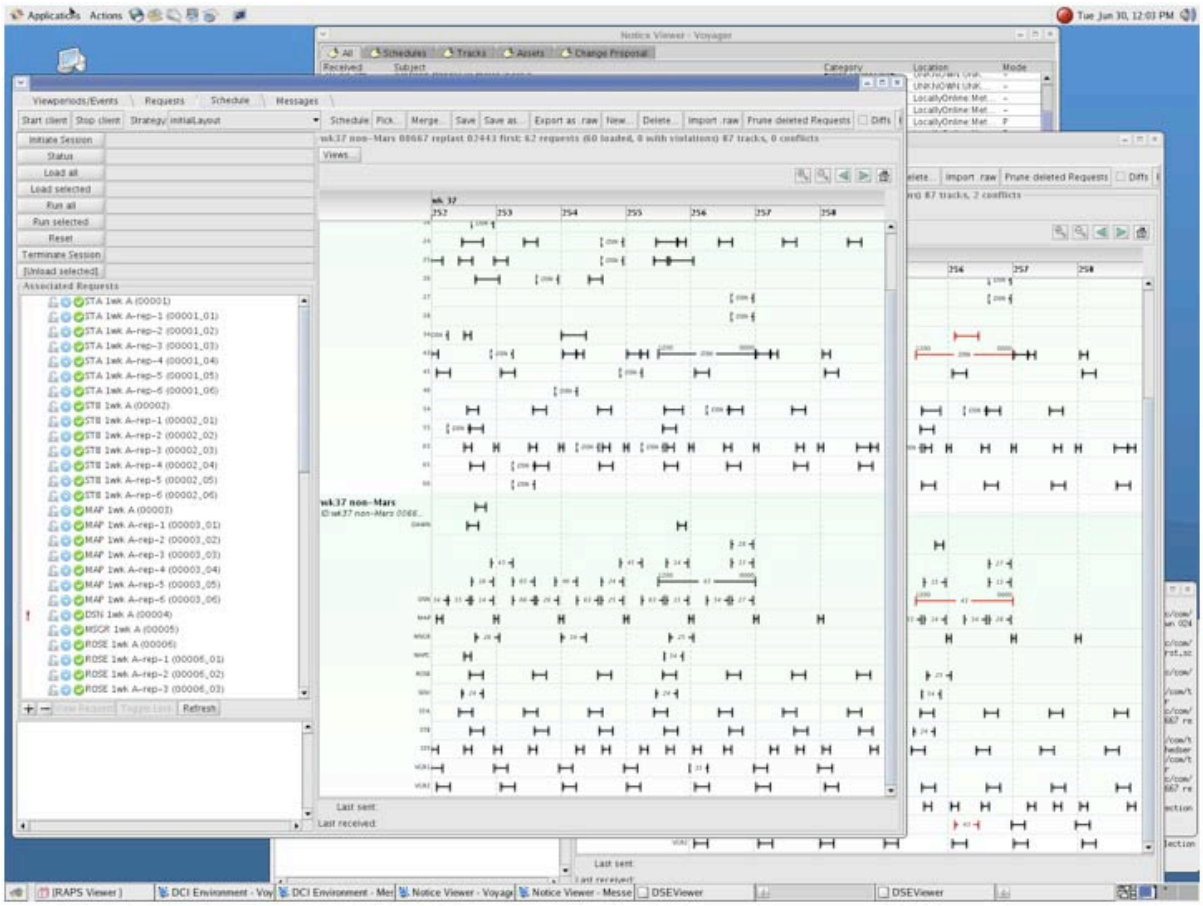

Figure 5 The schedule manager clicks on the launch buttons for notices resulting from applying the fallback request to view the conflict schedule and the fallback schedule side by side.

The 70-meter antenna at Canberra, DSS-43, must have some critical maintenance performed on it during schedule week 37. This asset change is broadcast by the SCS, received by both Voyager and Messenger agents, and both agents post notices of the asset change. But the Voyager agent's DSE service sees from its list of week 37 requests that it had previous requirements that used DSS-43 for that week. Based on a list of actions that the Voyager user rep allows the agent to take, the Voyager DSE Service first runs the DSE to prepare an initial schedule layout with the new DSS-43 maintenance requirement for week 37. (The DSE Service commands the scheduling engine and receives feedback on the results via a DSE-client script runner developed for this demonstration by Dr. Mark Johnston of JPL.) The initial layout results in conflicts and violations, so again, based on the list of user allowed actions, the DSE Service runs the DSE repair strategies for conflicts and violations.

At this point in the demonstration, the SCS sends out information about track changes that have resulted from the change in status of DSS-43. Each agent receives these changes, but only Messenger has tracks in the list. So the Messenger agent posts track change notices that include a button to launch the DSE-client to view those track changes (see Figure **4**).

After applying the normal repair strategies, there are still 2 conflicts in the Voyager schedule. The DSE service posts notices to that effect. From its request database in the SMS, the Voyager DSE Service knows the intent of the requirement causing the conflict as well as the allowed requirement reduction strategies, or fallbacks, for the Voyager 1 mission. Fallback strategies are not the minimums specified in normal requests, but are allowed by the mission in special circumstances. From these it determines that the requirement in conflict can be reduced for one day in this schedule week. So it prepares a new request and invokes the script runner to replace the original requirement with the new one. It also updates its request database with a link showing that fallback replaced the original request.

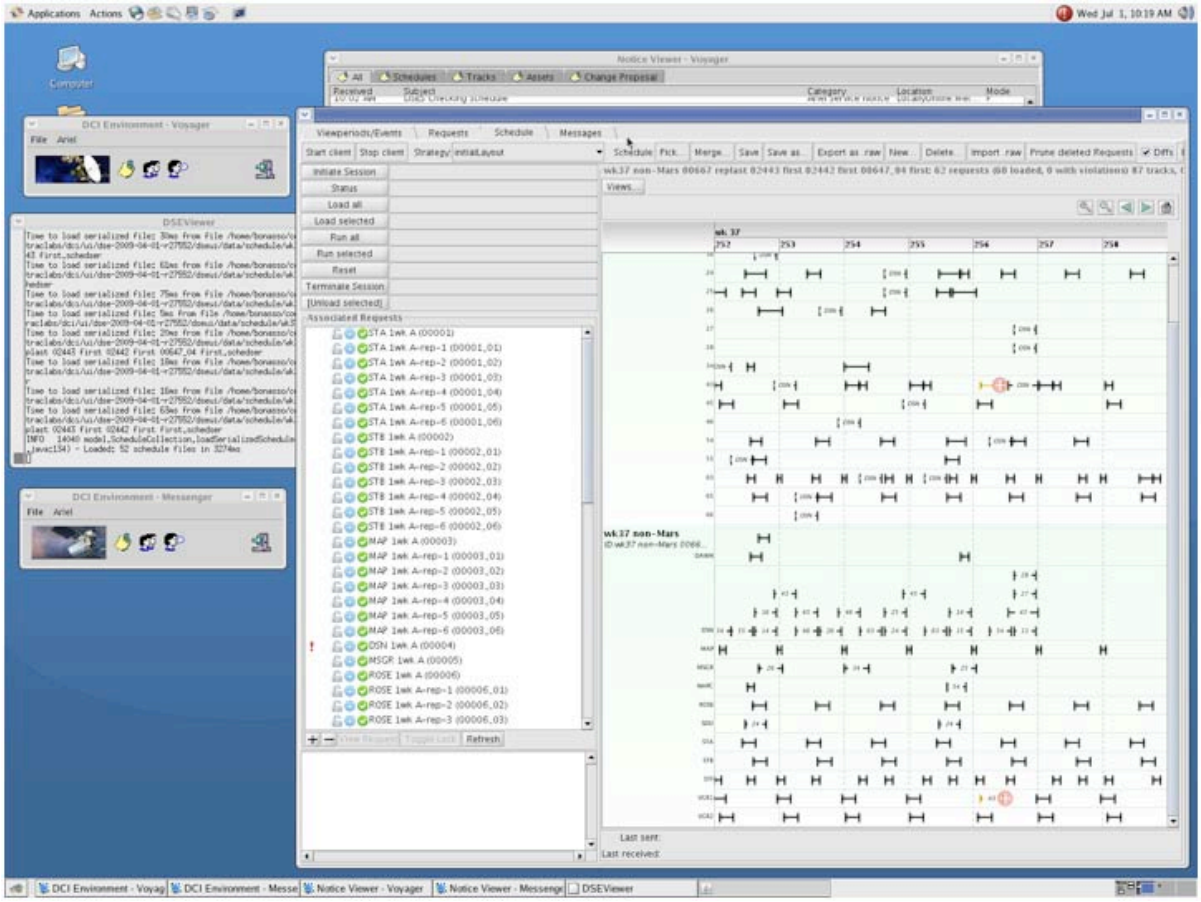

Figure 6 A successful schedule results from restoring the original request. A new notice appears to that affect, and the user rep launches the DSE-client from the launch button to view the new schedule. The user rep uses the diff button to see the restored track

Monitoring the results from the script runner, the DSE Service determines a problem-free schedule has been achieved and posts notices accordingly. The schedule manager views the notices, and from the notice buttons, launches the DSE-client for first the 2-conflict schedule, and then the schedule resulting from the fallback, viewing them side-by-side (Figure **5**).

### 4.2 Part two

In the second part of the demonstration, the SCS sends an asset change event indicating that the maintenance requirement for DSS-43 for week 37 has been reduced from 36 hours to 11 hours. Both the Voyager and Messenger agents post notices for their missions. Both DSE services search their request databases for requests with requirements in week 37 that use DSS-43.

The Voyager DSE Service locates the original request that was superseded by a fallback because of the loss of 70-meter antenna availability. So, based on its list of user allowed actions, the Voyager DSE Service executes a plan to delete the fallback request and add back the lost requirement. The results from the script runner are successful after the initial schedule layout (Figure **6**). As a last step, the DSE service removes the link between the fallback and the original requests, and once more makes the original request active in the SMS.

### 4.3 Demonstrated Capability

The demonstration showed how our software agent framework could significantly facilitate schedule management. First of all, the software demonstrated a number of functions of a DCI agent useful to the DSN missions. It showed that the DCI agent maintains user/mission status information, in particular the online/offline status of the user. It also showed how the DCI agent provides notices filtered and tailored to the supported mission, in particular, notices related to track

changes, asset and schedule changes, and change proposals.

The demonstration also showed how the agent state management service (SMS) can be extended to maintain mission and request status in order to support the action plans used by the DSE service.

As well, the software demonstrated several instances of launching $S^3$ application software, that is, the DSE-client, to display schedule information. This included schedules with conflicts, "diff'd" schedules to show track changes, and schedules from agent-modified requests, such as fallbacks and restored requests.

Most of the action took place within the DSE service of each DCI agent. This service monitored track, asset and schedule changes coming from the SCS, selecting only those relevant to the supported mission. It accessed locally stored schedule and request information to support required actions, and took action on the part of the mission to recover from an unanticipated loss of asset availability by first attempting repairs and then implementing a fallback strategy. Finally, it took action on behalf of the user rep to restore a modified request when the asset became available.

The SCS served as a focal point for generating events that might require action on the part of the user reps. By monitoring the master schedule data and sending change events, this service can preclude the user having to constantly pull information from the other side of the firewall. While we only demonstrated the messaging capability of the SCS, we plan to develop the complete functional design of the SCS in a follow on effort.

## 5. CONCLUSIONS & FUTURE WORK

This research showed that there indeed are a number of ways in which intelligent agent software can greatly aid the scheduling activities of the user scheduling representatives. In particular the agents can:

- Monitor around the clock for changes in assets and schedules
- Detect and notify user reps of relevant schedule events, including asset availability, request conflicts and changes that reflect unexpected opportunities to add or restore requirements for a given mission
- Execute the SE software in the background on the user's behalf by applying routine repair strategies, applying fallback strategies particular to the mission, and generating new requests that take advantage of new scheduling opportunities
- Though not completely demonstrated in the prototype, our discussions with JPL user reps indicated that DCI agents can aid in the multi-user negotiation process by posting coordination notices to affected user reps and by generating schedules from the proposed changes

Our prototype agent software demonstrated some part of every aspect of the above capabilities, and it did so by reusing our existing agent framework and by developing a new service – the DSES – tailored specifically to support DSN scheduling. Thus, it appears feasible to develop and deploy a comprehensive agent system to support the DSN user reps using DCI.

In a follow-on effort we propose to more fully develop the capabilities of the DCI agents for a selected set of JPL DSN user reps and to define a path for deploying DCI or similar agent capabilities in the future $S^3$ architecture. Additionally, we will investigate how useful agent technologies can be for the space (SN) and ground (GN) networks.

## 6. ACKNOWLEDGEMENTS

This work was supported by NASA SBIR grant NNX09CC16P. Dr Mark Johnston of JPL was instrumental in providing mechanisms for DCI to interface with the current S3 system.

## 7. REFERENCES

1. Clement, B.J. and M.D. Johnston, Design of a Deep Space Network Scheduling Application, in IWPSS 2006. October, 2006: Baltimore, MD.
2. Martin, C.E., et al. An Environment for Distributed Collaboration Among Humans and Software Agents. in 2nd International Conference on Autonomous Agents and Multi-Agent Systems. 2003. Melbourne, Australia.
3. Johnston, M.D., et al., Request Driven Scheduling for NASA's Deep Space Network, in IWPSS09. 2009: Pasadena, CA.
4. Firby, J.R., The RAPS Language Manual. 1999, Neodesic, Inc.: Chicago.