Proceeding of the 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space: i-SAIRAS 2003, NARA, Japan, May 19-23, 2003

When Automated Planning is Not Enough: Assisting Users in Building Human Activity Plans

Debra Schreckenghost, Mary Beth Hudson, Carroll Thronesbery, and Kevin Kusy
NASA Johnson Space Center
TRACLabs
1012 Hercules
Houston, TX 77058
{d.schreckenghost@jsc.nasa.gov, hudson@traclabs.com, c.thronesbery@jsc.nasa.gov, kkusy@sk-tech.com}

Keywords Automated Planning, Crew Activity Planning, Assistive Software

Abstract

For long duration, manned space operations to be cost-effective, it is necessary to provide tools that reduce the human workload for both space-based and Earth-based operations. Automated planning software has the potential to reduce human workload for planning human activities such as crew activities. An important challenge is assisting people in using automated planning software to construct and monitor the execution of crew activity plans. We have addressed the problem of making automated planning accessible for activity planning by developing new software to assist people in using existing automated planners. We have developed a *planning assistant* that helps a user understand how a plan input specification results in a particular plan and assists the user in changing this specification to produce a plan better suited to the user's needs. Important features of the planning assistant include the integration of planning and scheduling techniques, support for user evaluation of plans, and management of multiple plan versions. The planning assistant includes integration of these advanced planning techniques with standard information technology like databases and web-based interfaces.

1. Challenges in Using Automated Planners

For long duration, manned space operations to be cost-effective, it is necessary to provide tools that reduce the human workload for both space-based and Earth-based operations [2]. Automated planning software has the potential to reduce human workload for planning human activities such as crew activity plans. An important challenge is assisting people in using automated planning software to construct and monitor the execution of crew activity plans. Most automated planning software today has limitations that constrain its utility for planning human activity. These limitations are described below [5]:

- *Understanding and influencing how a plan input specification results in a particular plan or in an inability to plan*
  For most planners, the specification of the goals and situation information that change with each version of a plan is accomplished using either text files or simple menu-based systems that manipulate predicate lists. Using such interfaces, it is difficult to understand how a plan input specification results in a particular plan or what to change to improve a plan.
- *Adjusting a plan based on user preferred temporal and resource constraints*
  Building activity plans for humans working in space requires the ability to provide a rich constraint language for temporal placement and resource usage of tasks. These constraints include preferred and exact temporal placement of activities, designated critical resources, and complex resource modeling (pooled, dedicated, profiling). Few automated planning systems can represent and utilize such constraints.
- *Managing multiple versions of a plan*
  Planning space activities requires building alternative plans for important anticipated contingency situations. The current approach to automated planning requires the user to manage the different variations on input that generate different versions of the plan and provides no support for evaluating and comparing the resulting plans.
- *Accessing planning information remotely*
  Recent developments in web and database technology have improved the ability to remotely access information. Most automated planners do not take advantage of these technologies.

2. Approach for Assisting Users

We have addressed the problem of making automated planning accessible for activity planning by developing new software to assist people in using existing automated planners. We have developed a *planning assistant* that helps a user understand how a

plan input specification results in a particular plan and assists the user in changing this specification to produce a plan better suited to the user's needs. It provides plan management capabilities, such as

- Support for generating, comparing, and storing multiple versions of a plan,
- Computation of plan metrics that aid in comparing plans,
- User control over how tasks are planned based on task priority, resource usage, and temporal constraints associated with goals,
- Use of a database to store input specifications for the planner as well as output descriptions produced by the planner, and
- Use of a web interface to remotely view and modify planning information in a database.

To provide such planning capability, we have combined planning and scheduling techniques using domain-specific knowledge for the purposes of planning human activities. We use hierarchical planning to reason about user goals to determine what tasks to do. We extend the task representation and constraint matching in the hierarchical planner to model complex constraints including pooled and dedicated resources, resource budgets, temporal constraints expressed in absolute time (e.g., schedule meeting at 2:00 on Tues), and user preferred timing constraints that can be relaxed if needed. We use scheduling algorithms to select top-level planning goals from a set specified by the user. These user goals have associated constraints (priority, temporal, resource) that are used both in selecting goals for planning as well as matching constraints during goal decomposition. Since the selection and ordering of goals from the user for planning is interleaved with goal decomposition, we can successfully build plans where a subset of the user-specified goals is achieved. When no top-level goal meets scheduling criteria, a block of unscheduled time (called *free time*) is added to the plan. By aiding the user in adjusting which scheduling criteria are considered, the order in which top-level goals are planned changes, which results in different versions of a plan to meet these goals.

3. Description of the Planning Architecture

Our approach is to develop a planning toolkit consisting of a set of distributed processes that interact to provide the integrated planning capability illustrated in Figure 1. The modules of the toolkit are listed below:

- **Planning Assistant**: This software assists the user in incrementally building an activity plan and provides for interim plan evaluation, comparison, and modification. It aids the user in specifying activity preferences, evaluating trade-offs between different versions of plan, and understanding the implications of these choices on the resulting plan. It provides capability for the user to monitor the execution of the activity plans as well.
- **Planning Software**: Hierarchical planning software modified to include scheduling algorithms and complex resource modeling for the purposes of planning human activities.
- **Planning Database**: Database storing plan versions. A version of a plan consists of seven tables describing the plan input specification and the plan output generated by the planning software. Two tables of application knowledge are used by all plan versions.
- **Web Interface**: The web interface provides remote access to the planning information stored in the planning database. The web interface is used to view the results of planning, to monitor the execution of plans, and to manually mark tasks complete.

In this paper we describe our planning assistant software that assists users in interacting with automated planners and our approach for integrating this assistant software with an automated planner, a database, and web interfaces.

4. Combining Planning and Scheduling

Traditionally, planning research has not addressed sophisticated temporal or resource modeling, nor provided support for optimizing solutions [7]. Yet these capabilities are needed for crew activity planning. We have modified AP [3], a HTN that includes some temporal modeling such as expected task duration and temporal ordering constraints based on Allen's temporal logic [1] to include scheduling features such as complex resource modeling, user preferred temporal constraints, and optimization criteria. Our approach differs from sequential integration approaches that view scheduling as a post-planning phase [8] in that planning and scheduling functions are tightly coupled and dependent upon each other to produce plan. Our approach in similar in principle to the integrated approach described in [4] that use planning to determine actions and scheduling to manage constraints. We use the planner to plan the overall week in a crew schedule, but we use the scheduler to help in selecting the individual tasks performed each day of that week. To do this, we have modified the hierarchical task net planner to include scheduling software that dynamically selects the best goals from a set of desired goals based on temporal and resource constraints. Since these criteria are under user control, the concept of "best" can be changed to suite different circumstances. As a result, not all goals have to be achieved for a plan to be built, although the resulting plan may be sub-optimal. This approach also makes it possible to define planning metrics that can be used to adjust scheduling criteria and improve plans.

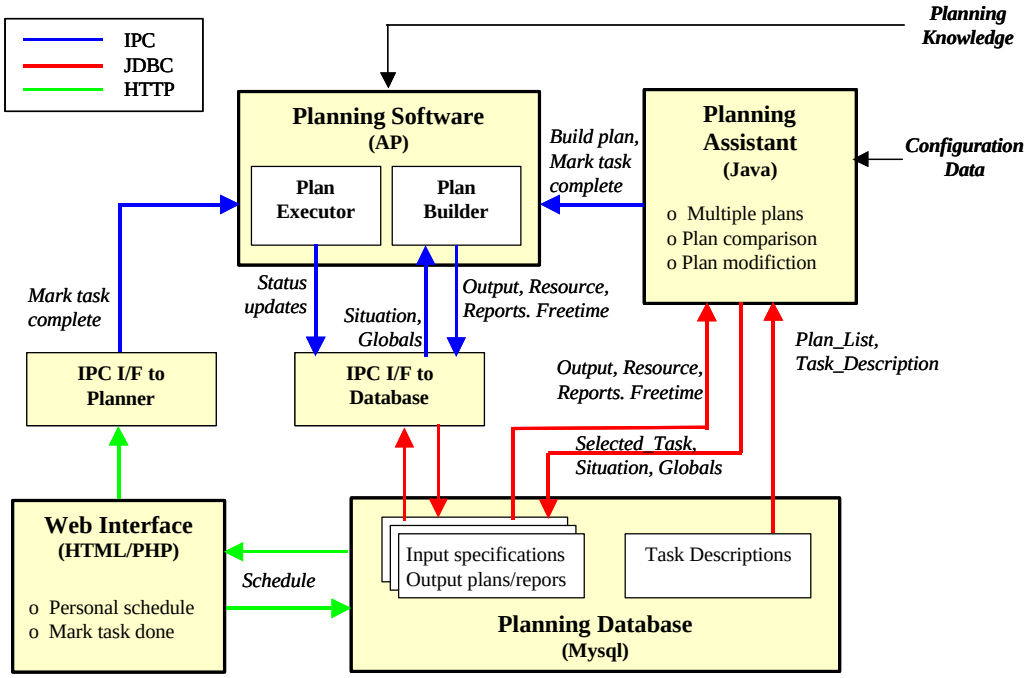

Figure 1. Architecture of the Planning Toolkit

We have selected scheduling criteria to match crew needs. For crew activity plans, it is important to be able to schedule tasks at specific times. We have implemented temporal constraint handling algorithms that model both required timing and user preferred timing for goal achievement. Crew activities require the use of both dedicated and pooled resources. Thus, we have implemented resource handling algorithms that model both these types of resources, and that distinguish critical resources from non-critical resources. Finally, goal priorities can change in response to changing circumstances. We have implemented priority handling algorithms that model goal priority that can be optionally considered when selecting goals. Crew activity goals are selected using the following optional criteria:

- Prefer the goal with the maximum expected duration that fits within the allocated time
- Require the predicted power usage of the goal to fit within the available power budget
- Require hard temporal constraints to be satisfied for the associated goal to be planned
- Prefer team goals before individual goals
- Prefer high priority goals over low priority goals
- Satisfy user preferences if possible

These criteria are based on typical techniques used by scheduling software when placing tasks to meet temporal and resource constraints.

These planning and scheduling capabilities are used in the following manner. The planner uses hierarchical goal decomposition to build a daily sequence of tasks based on templates describing typical operations for weekdays, weekends, and EVA days. These templates include blocks of time during which the crew can accomplish mission goals (called *duty blocks*). The tasks scheduled during these duty blocks vary daily, while the other blocks in the template are invariant for the template. By blocking out time in this manner, we only need to concern ourselves with the details of the duty blocks because these are the times when there are conflicts for resources and temporal constraints. We do not need to schedule the details of the other blocks. We leave it up to the crew to manage the details during the other blocks, thus allowing them some amount of autonomy over their schedules. Figure 2 shows an example of a daily planning template.

Weekday Plan Template	
morning block	**2.0 hrs**
status with Earth	**0.5**
morning duty block	**3.0**
meal block	**1.0**
afternoon duty block	**4.0**
planning meeting	**0.5**
exercise block	**2.0**
evening block	**2.0**
sleep block	**9.0**

Figure 2. Example of Daily Planning Template

The planner uses scheduling criteria to help in selecting the goals to be performed during the daily duty blocks. It does this by evaluating the set of goals to be planned with respect to the scheduling criteria that have been enabled by the user. The scheduler makes sure that temporal constraints, resource constraints, and user preferences (if enabled) are met when selecting goals to be planned. Once a goal has been selected, the planner then uses hierarchical goal decomposition to further break down the selected goals into sub-goals, conditional upon the current situation. Finally, the planner identifies operations to achieve these goals, assigns agents to perform these operations, and manages the temporal ordering for any subtasks in the operation.

Each goal to be planned may have associated with it a priority, either a hard constraint or a user preference as to when it should be done, and one or more required resources. These characteristics are used by the scheduler when determining which goals should be planned next. We describe our scheduling approach in the remainder of this section.

The basic mechanism of the scheduler is as follows. Given a specified time, the duration of the duty block, and a set of goals to be performed, the scheduler first determines which goals are possible candidates at the specified time. That is, it selects all goals whose temporal constraints, resource constraints, and user preferences are met at the time specified. Given this list of candidate goals, the scheduler then determines which goal ranks highest in the list. To determine this, the scheduler looks at the optimization criteria set by the user. The optimization criteria are evaluated in a user-specified order and consist of the following categories:

- Meet required temporal constraints
- Meet user-preferred temporal constraints
- Prefer high priority goals
- Prefer goals with longer expected duration
- Prefer goals using critical resources

When the scheduler is ready to select among the candidate goals for a given block of time, the candidate goals are sorted based on these criteria in the order of importance. The scheduler selects the highest ranking goal based on the selected criteria and passes this goal to the planner. The planner further decomposes the goal into primitive tasks and updates the amount of time that is available in the duty block. The scheduler then re-computes the candidates based on the effects that the scheduled task has on the schedule, e.g., resource usage, temporal changes, etc. The cycle repeats, with candidate goals being identified, the highest ranking goal being selected and the remaining time being returned for scheduling, until the duty block is filled or there are no tasks which fit in the block of time being scheduled. If the scheduler fails to find a goal that fits in the duty block, then the planner plans free-time for the remainder of the block. The planner then continues planning the day until all duty block are filled. Because the planner schedules free-time whenever a duty block cannot be filled by an outstanding goal, the planner never fails to build a plan. Instead, if no goals can be met, the crew's work week is filled with free-time. So the amount of free-time in a plan is one indication of the "goodness" of the plan. If a plan contains too much free time, the user can always specify "minimize free-time" as an scheduling criteria which will cause the scheduler to try to pack the plan as full as possible, overriding other considerations, such as priority of goals, etc.

Since goals are selected on a first fit basis, the order that we consider the goals is significant. Given a block of time to be planned, all goals that fit within that block of time are candidate goals. To determine which one of the candidate goals is passed to the planner, the scheduling algorithm looks at optimization criteria specified by the user and orders the tasks by these criteria. The user specifies the characteristics that should be maximized for a given plan as part of the input to the planner. This allows the user to create several variations of a plan whose initial conditions and goals are the same, but whose scheduling criteria are different.

We describe the scheduling criteria using a language that characterizes features the user desires in the resulting schedule. We then translate these descriptions into parameters for the optimization routine. For example, the user may want a schedule that is packed (i.e., minimizes free-time). Or she might want one that meets as many crew preferences as possible. Or she may want all the highest priority tasks scheduled. All of these user constraints can interact with each other. Taken alone, each criteria would produce a different schedule.

We allow the user to select and rank each criteria which she deems is important in producing a plan. Thus, more than one criteria is considered when producing plans. We also provide the user with the opportunity to adjust the plan by changing the order of importance of the scheduling criteria. The candidates are then sorted by each criteria using this order of importance. To implement this we use a stable sort and sort the candidates based on the least important criteria first, then sort by the next-to-least important criteria, etc. until we sort by the most important criteria. Then the first goal in the resulting sorted list is selected to be planned.

The scheduler models four types of constraints: temporal preferences, temporal constraints, priority, and resources. Temporal preferences and temporal constraints are very similar, the difference being that a temporal preference says that a task should be done within a specified window if possible, but it is feasible to do it outside that preferred window. A temporal constraint says that a task must be done within a certain window or not done at all. Goal priority is modeled simply as an integer denoting the relative importance of the tasks. The most complex constraint model is the model of resources used to

determine when all the necessary resources are available for the task to be done. The complexity of the scheduler's job is based on what types of resources it manages. In our scheduler we model both *dedicated* and *pooled* resources. We also model resources as consumable (i.e., can be produced or consumed) or non-consumable. In our application, the quantity of resource used is fixed for the duration of the task. If a resource is produced or consumed, that resource is modified at the scheduled start time of the task.

Pooled Resources: A *pooled resource* is a resource that may be used by multiple tasks at the same time. Therefore, there is some quantity associated with it denoting how many units there are available. Examples of a pooled resource might be power, oxygen, or water. The initial quantity of a resource must be specified prior to building a plan. A task would specify how much of the resource it requires and for how long. Pooled resources are modeled using lists of segments - where a segment is a start and end time and a quantity. For example, (0 2 qty 5) means a resource quantity of 5 starting at time 0 and ending at time 2. The start and end times are relative to some fixed point in time known outside of the scheduler. In our application, the time units are hours, but any time unit can be used as the scheduler simply represents them as integers with no units. This makes the scheduling engine very robust in being able to handle any unit of time for any time horizon.

Dedicated Resources: A *dedicated resource* can be considered a pooled resource whose quantity is always one or zero. It is significantly more efficient to model it differently, however. An example of a dedicated resource might be a person or a tool. Dedicated resources cannot be shared among multiple tasks and a task uses all of the resource assigned to it. Dedicated resources are modeled using lists of intervals – where an interval is a start and end time, and the list of intervals designates the times that the resource is available. For example (0 2) (4 6) (7 10) means that a resource is available (has a qty of 1) during the times 0 to 2, 4 to 6, and 7 to 10, and the rest of the time it is unavailable. Like pooled resources, the times are relative.

Consumable Resources: The scheduling engine provides the capability for resources to be produced, consumed, or reused. For example, a task might produce a resource such as oxygen. Another task might consume a resource such as power. Or, a task might use a resource and at the end of the task return that resource (e.g., a hammer). These types of behaviors can be used with both the dedicated or pooled resources. All consumable resources must have a quantity defined for the duration of the scheduling window, where the scheduling window is the total time period that we are planning over. The quantity of a consumable resource may be zero for a time interval.

Another capability we have added to the planning software is the ability to plan *periodic tasks*. Periodic tasks are tasks that must be performed on some regular time interval. The concept of repeatedly accomplishing the same goal is not well-suited to hierarchical planners, since once a goal is accomplished there is no natural mechanism for making it become available for planning again after a time period. We use the goal-handling mechanisms developed for applying scheduling criteria to implement periodic tasks. At the beginning of planning each day, we compare the current day to the day the periodic task was last performed. If the elapsed time is greater than or equal to the defined period, the goal is added back to the list of goals to be planned.

5. Evaluating a Plan

The planning assistant helps the user evaluate the "goodness" of a plan and compare multiple versions of a plan using this assessment of goodness. To evaluate plan goodness, the assistant provides plan statistics that characterize how many of the requested goals were actually scheduled, what types of goals were scheduled, and how efficiently these goals were scheduled. The types of goals distinguished in these statistics include (1) goals with priorities, (2) goals with preferred timing constraints, (3) goals with required timing constraints, and (4) goals with required resources. Plan efficiency is measured by determining the amount of duty time that is unscheduled (i.e., *free time* refers to a period of duty time where no mission activity is scheduled).

Versions of a plan can be compared from two perspectives:

- Differences in the plan input specification information used to generate the plan
- Differences in the resulting plan generated from this specification

The planning assistant provides the user with support for comparing plans from both these perspectives. It does this by providing metrics about the plan that characterize both the input specification and the output generated by the planner. The values for each metric are then compared across the chosen plans and highlighted in the plan with the best setting (i.e., the maximum number of goals scheduled or the minimum unscheduled time).

The following metrics are provided for comparing the plan input specification information across plan versions:

- **Goal Comparison**: how are the specified goals different among versions? This is supported by comparing the lists of selected goals for each plan and the timing and priority settings for each selected goal.
- **Goal Scheduling Criteria:** how does the planner determine which of the specified goals should be chosen for planning. This is supported

by comparing the settings for the scheduling criteria associated with each version of the plan.

The following metrics are provided for comparing the generated planning outputs across plan versions:

- **Goal Achievement**: does the plan include tasks to achieve all user goals? This can be evaluated by reviewing the percentage of goals scheduled and not-scheduled in the *all-tasks* column of the plan statistics view.
- **Compliance with User Preferences**: does the plan comply with all user preferences? This can be evaluated by reviewing the percentage of goals scheduled and not-scheduled in the *preferences* column of the plan statistics view.
- **Goal Priorities:** are all high priority goals achieved? what low priority goals are not achieved? This can be evaluated by reviewing the percentage of tasks scheduled and not-scheduled in the *priorities* column of the plan statistics view.
- **Plan Efficiency:** What is the percentage of unscheduled time (called free time) in plan? This can be evaluated by reviewing the percentage of tasks scheduled and not-scheduled in the *free time* table of the plan statistics view.
- **Resource Utilization:** What is the predicted usage profile for the plan? This can be evaluated by reviewing the percentage of goals scheduled and not-scheduled in the *resources* column of the plan statistics view.

For all these metrics, the user can point to a percentage and see a list of the goals associated with the statistic (e.g., a list of the goals with preferences that were scheduled).

These plan metrics can be used to determine how to improve a plan. The following summarizes typical problems with a plan and how to respond to them using the planning assistant.

- Too many unplanned goals: Minimize free time
- Priority goals not scheduled: Prefer high priority or adjust priority settings
- User preferences not met: Enforce preferences or adjust preferred timing settings
- Goals with hard timing constraints not scheduled: Adjust required timing settings
- Critical resources under-utilized: Maximize resources or change the list of critical resources

If the plan has more than one of these problems, multiple criteria can be selected and ordered according to importance. If adjustment of the scheduling criteria does not result in an acceptable plan, the user should consider adjusting the timing and priority settings for the goals not scheduled.

Typically, the user first creates a schedule where all optimization criteria have equal weight. After examining the metrics for this plan, the user may want to adjust which criteria are considered or the order in which they are considered. For example, if the first plan contains too many unplanned goals, the user can rank the minimize-free-time criteria as the most important (i.e., the first to be evaluated). When selecting more than one criteria, the user specifies the order of importance among them by numbering them sequentially. The user can then produce another plan and compares it to the first plan using the planning metrics provided.

6. Managing Versions of the Plan

Planning space activities requires building alternative versions of plans. These alternatives arise when evaluating trade-offs in user preferences and task priorities, or when evaluating the effects of possible contingency situations. Each version of a plan corresponds to a unique set of information used to initialize the planner (i.e., the initial state of the environment and the goals to be achieved by the plan). Most automated planners, including the planner we used, can only reason about one set initialization information and one resulting plan at a time. This requires the user to manage by hand the variations on planning input that generate different versions of the plan.

The planning assistant aids the user in managing versions of a plan by storing both the plan input specification and the results of plan building in a database. A version of a plan consists of the following seven database tables that store attributes of the version, an input specification for building a plan, and the resulting planning products:

- **Selected Tasks**: Goals selected by the user for this version of the plan.
- **Situation Table**: Initial world state and conditions used for this version of the plan. These conditions include the selected goals, the initial states, and information from the Task Description table describing the selected goals.
- **Global Table**: Scheduling criteria specified by the user for this version of the plan. These criteria include which scheduling criteria should be used and the evaluation order for these criteria (which indicates relative importance).
- **Output Table**: Time-ordered tasks assigned to specific crew for this version of the plan
- **Report Table**: Statistics about how well this version of the plan meets its scheduling criteria. These statistics consist of percentages describing how many goals in a report category are planned, and how many in the category are not planned.
- **Free-time Table**: Distribution of free-time in the plan. This table shows the free-time assigned to each crew member for each day of the plan
- **Resources Table**: Predicted resource usage and available resources for the plan. The available resources are loaded into the planning assistant as part of the startup configuration file. The predicted resource usage is computed by

summing the required resources specified for each task planned.

An instance of these seven tables is created for each version of the plan.

The planning assistant indexes each set of seven tables to a version identifier that is stored in a master list of plans table (**Plan List Table**). It handles storing and updating these tables when the user changes planning information or builds a new plan. It also retrieves planning information from these tables based on the version identifier, and can command the planner to regenerate the associated plan output and update it in the database. Attributes of the tasks selected for planning are retrieved from a **Task Description Table** that is independent of a particular version of the plan, which permits defining tasks that are used repeatedly across plans. Multiple plan versions can be loaded into the planning assistant at one time, and the user can easily interact with these different versions from the Planning Assistant user interface. The input specification tables from a version also can be copied as a starting point for building a new plan, permitting the user to carry over routine goals, states, and settings from week to week.

An important aspect of plan management is characterizing what is unique about each version of the plan. The planning assistant provides succinct views of the planning information that make it easy to differentiate plan versions. This information includes a summary of version attributes (such as start time, plan length, and version description), the selected tasks with their priorities and temporal constraints, and the chosen scheduling criteria. It also provides metrics about each plan that can be compared across versions to aid in differentiating them.

7. Integrating with Web Technology

When planning space activities for long duration missions, it is important for crew to have ready access to personal schedules and to monitor progress made on accomplishing tasks in these schedules. They need to understand which tasks are in progress and which tasks have been completed. Such information should be available to them from any location in the space facility. Web-based interfaces can provide such remote access to crew schedules.

The use of a database to manage planning information makes it straightforward to include web-based technology as part of the planning toolkit. We use web interfaces to view the results of planning, to monitor the execution of plans, and to manually mark tasks complete.

The Planning Toolkit makes two plan views available from a web browser. One view shows the entire plan for all crew in a tabular form. A second view provides each crew member with a view of their personal schedule. This second view was designed to fit on the display of a handheld computer. Information on the personal schedule includes the date, the start time, the task, and the estimated length of the task. Figure 3 shows an example of a crew personal schedule.

From this view of his personal schedule, a crew member also can monitor the completion status of tasks and be reminded of the next task to be performed. Since this display is based on information in the database, updates to task completion status are accessible by monitoring for changes in the database. As shown in Figure 3 below, completed tasks are shown with a darker gray background.

JOHN - Nominal_May_27_2002_Output

Mon May 27	06:00 personal_time_in_morning	2 hr
	08:00 daily_status_with_earth	0 hr
	08:30 recharge_robotic_batteries	3 hr
	11:30 lunch_during_the_work_day	1 hr
	12:30 administer_monthly_medical_exam	2 hr
	14:30 receive_monthly_medical_exam	2 hr
	16:30 daily_exercise_period	2 hr
	18:30 daily_planning_meeting	0 hr
	19:00 personal_time_in_evening	2 hr
	21:00 sleep_period	9 hr
Tue May 28	06:00 personal_time_in_morning	2 hr
	08:00 daily_status_with_earth	0 hr
	08:30 prepare_to_service_antenna	2 hr
	10:30 service_antenna	6 hr
	16:30 post_service_operations	2 hr
	18:30 daily_planning_meeting	0 hr
	19:00 personal_time_in_evening	2 hr
	21:00 sleep_period	9 hr

Figure 3. Example of Crew Personal Schedule

Current Status: recharge_robotic_batteries

schedule:	Nominal_May_27_2002_Output
task:	recharge_robotic_batteries
performed by:	JOHN
execution status:	planned

Procedure: recharge_robotic_batteries

Currently, there are no instructions to support recharge_robotic_batteries operations. You can still verify the execution status of the task.

completed | Enter Information

[top] [personal schedule]

Figure 4. Crew Marking Task Complete

The web-based personal schedule can be used to mark tasks complete. The name of each task is an active link to a web page where the user can enter task completion status manually. Once the status has been entered, the status is updated and the web

browser returns to display the personal schedule. Figure 4 shows an example of a web page where tasks are marked complete.

The Planning Toolkit provides two different approaches for marking task completion status from a web page. In one approach, marking a task complete sends a message to inform the automated planner that the task goal state has been achieved. The planning software then recognizes that the associated task is complete, and stores the changed status in the Output database table. This approach is needed for closing the loop on automated planning, and makes it possible for the planner to dynamically re-plan when tasks fail to complete. In the second approach, marking a task complete directly updates the Execution Status field in the Output database table. This approach takes the automated planner out of the plan execution loop, and supports operations where the planner is only being used to build plans.

Our approach for retrieving information in the database using a web browser is to use PHP scripts that construct the database queries and generate HTML code from the results of these queries. Our approach for updating information in the database from a web browser is to use a cgi-bin executable that activates a script to send commands to the automated planner via IPC (the client-server communication protocol from Carnegie-Mellon University used by the planner; [6]). This approach required access to the cgi-bin directory on the web-server machine and required an IPC Central Server process to be running.

8. Conclusions

We have developed a Planning Toolkit to assist users in developing crew activity plans. This toolkit includes a Planning Assistant that mediates between the user and automated planning software. This assistant helps a user understand how a plan input specification results in a particular plan and assists the user in changing this specification to produce a better plan. It provides plan management capabilities, such as

- Support for generating, comparing, and storing multiple versions of a plan
- Computation of plan metrics that aid in comparing plans
- User control over how tasks are planned based on task priority, resource usage, and temporal requirements or preferences
- Information displays for both input specifications and plan outputs

To provide such flexible planning, we modified existing automated planning software. New capabilities added include modeling complex temporal and resource constraints and adding scheduling software to select the "best" high-level goals to plan. We allow the user to select and order the criteria used by this scheduling software as a means of producing alternative plans for the same set of goals. To make the toolkit more usable, we integrated these advanced planning techniques with standard information technology like databases and web-based interfaces.

9. Acknowledgements

This research was funded by NASA at the Johnson Space Center under the Small Business and Innovative Research (SBIR) program.

10. References

[1] Allen, J. F., Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26, p 832-843.

[2] Cooke, D., and B. Hine. NASA's New Era in Space Exploration. *IEEE Intelligent Systems*, March/April, 2002. 17(2).

[3] Elsaesser, C., & MacMillan, T.R. Representation and Algorithms for Multiagent Adversarial Planning. Technical Report MTR-91W000207. MITRE, Wash. D.C. 1991.

[4] Garrido, A., and Barber F. Integrating Planning and Scheduling. *Applied Artificial Intelligence*. An International Journal. (ISSN: 0883-95-14). Taylor & Francis Eds. (2001).

[5] Schreckenghost, D., and Hudson, M. B. Automation in Context: Planning Manned Space Exploration Activities. *ISAIRAS 2001*.

[6] Simmons, R. and Dale, J. 1997. *Inter-Process Communication: A Reference Manual*. *IPC Version 6.0*: CMU Robotics Institute

[7] Smith, D., E., Frank, J., and Jonsson, A. K.. Bridging the gap between planning and scheduling. Knowledge Engineering Review, 15 (1), 2000.

[8] Srivastava, B., and Kambhampati, S., Efficient planning through separate resource scheduling. In *Proceedings of AAAI Spring Symposium on Search Strategy under Uncertain and Incomplete Information*. 1999.