Using ASPEN to Automate EO-1 Activity Planning

Rob Sherwood, Anita Govindjee, David Yan,
Gregg Rabideau, Steve Chien, Alex Fukunaga
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109
aspen@aig.jpl.nasa.gov

Abstract--- This paper describes the application of an automated planning and scheduling system to the NASA Earth Orbiting 1 (EO-1) mission. The planning system, ASPEN, is used to autonomously schedule the daily activities of the satellite. The satellite and operations constraints are encoded within a software model used by the planner. This paper includes a description of the planning system and the associated modeling language. We then discuss how we encoded the EO-1 spacecraft operations with the modeling language. We conclude with a description of the end-to-end planning system as we envision it for EO-1.

TABLE OF CONTENTS

1. INTRODUCTION

Automated planning/scheduling technologies show great promise in reducing operations costs by increasing autonomy of EO-1 mission operations. The Artificial Intelligence (AI) Group at the Jet Propulsion Laboratory has been working on a system called ASPEN (A Scheduling and Planning Environment). ASPEN [3] is a modular, reconfigurable application framework based on AI techniques [1], which is capable of supporting a variety of planning and scheduling applications (similar to [4]). The primary application area for ASPEN is the spacecraft operations domain.

EO-1 [5] is an Earth imaging satellite to be launched in May 1999. The science payload on EO-1 is an advanced multispectral imaging device. Mission operations on EO-1 consist of managing spacecraft operability constraints such as power, thermal, pointing, buffers, consumables, and telecommunications. EO-1 science goals involve imaging of specific targets within particular observation parameters. Managing EO-1 spacecraft downlink is particularly difficult because the amount of data generated by the imaging device is quite large and ground contacts are limited. In addition, because science targets for EO-1 are based on short-term cloud predictions, schedules must be generated daily.

Planning and scheduling spacecraft operations involves generating a sequence of low-level spacecraft commands from a set of high-level science and engineering goals. ASPEN encodes spacecraft operability constraints, flight rules, spacecraft hardware models, science experiment goals, and operations procedures to allow for automated generation of low-level spacecraft sequences. By automating the command sequence generation process and by encapsulating the operations specific knowledge, ASPEN will enable the EO-1 spacecraft to be controlled by a small operations team and thereby reduce costs.

2. ASPEN

ASPEN is an object-oriented system that provides a reusable set of software components that implements the elements commonly found in complex planning/scheduling systems. These include:

- An expressive constraint modeling language to allow the user to define naturally the application domain
- A constraint management system for representing and maintaining spacecraft operability and resource constraints, as well as activity requirements
- A set of search strategies
- A temporal reasoning system for expressing and maintaining temporal constraints
- A graphical interface for visualizing plans/schedules (for use in mixed-initiative systems in which the problem solving process is inter-active

We have implemented the linear programming simplex algorithm to optimize schedules for ASPEN. At present, though, ASPEN is not making use of it. There are hooks in the code to add this and other optimization algorithm at a later date.

The central data structure in ASPEN is an activity. An activity represents an action or step in a plan/schedule. An activity has a start time, an end time, and duration. In

```
1     Activity ALI_data_take {
2        Fixed fi;
3        Tracking tr;
4        Duration = [1,60];
5        Constraint =
6            starts_after end of SAD_changer with (fi->sad1) by [100,300] ,
7            ends_before start of SAD_changer with (tr->sad1) by [16,16] ,
8            Contains SAD_user with (fi->ap1) by [4,4,0,10],
9            Contained_by SAD_user with (fi->sad1) by [300,300,1,161;
10       Subactivities = ALI_user_ data, ALI_dark_count;
11    reservat.ions = processor, array_power = 80;
12    };
13
14    Activity SAD_changer {
15       Sad_mode sad1;
16    reservat.ions = solar_array change_to sad1 ,
17                    aperture must_be open;
18    };
```

Figure 1: Activity Example

addition, activities can use one or more resources. For more details on ASPEN, scc [2,3].

3. MODELING LANGUAGE

The ASPEN modeling language allows the user to define activities, resources, and states as described above. A domain model is input at start-up time, so modifications can be made to the model without requiring ASPEN to be recompiled. The modeling language has a simple syntax, which can easily be used by spacecraft mission operations personnel to create a model. Each spacecraft model is comprised of several plain-text files, which define and instantiate activities, resources, and states.

Activities

As previously mentioned, activities arc the central data structure of ASPEN. An activity is a data structure that performs a specific function. The example in Figure 1 includes an instrument data take activity and a solar array drive state change activity. These examples will be used to explain the components of an activity.

An activity is defined in line I and 14 of Figure 1. The definition includes the name followed by a pair of braces and a semi-colon similar to the C language syntax. These arc the only required components of an activity definition. Once the activity is defined, it can be instantiated in the initial state file. Generally, this instance will consist of the activity name followed by the instance name and a pair of braces. Many of the components below that are specified as ranges can be fixed to specific values in the activity instance.

Parameter-s arc generally used to store values in activities or reservations. Lines 2 and 3 contain parameters defined elsewhere in a parameters file. In this case, they arc constants that represent state names. Parameters can also be passed into activities from higher level activities (parent activities). Line I 5 contains an example of a parameter that is pawed into an activity. The parameter sad_mode is an enumerated type variable that contains the current state value of the solar array drive. Any one of the state values can be passed into the SAIJ_.changer activity when called from a parent activity.

The duration of an activity is given as a range [x, y], a list {a, b, c, d } or a constant Line 4 defines the duration as a range of 1-60 seconds. The time scale of the spacecraft mission planning can also be specified. All ranges within ASPEN can be specified from negative infinity to infinity. If a range is given for the duration, ASPEN will have more flexibility in considering different schedules. "I'his can result in better-optimized schedules.

Constraints arc temporal constraints an activity must satisfy wi(h respect to the (owner) activity in which the constraint is defined. There are six types of constraints: starts_before, starts_after, ends_before, ends_after, contains, and contained by. The first four constraint types include a time range and a temporal relationship to the start of or cad of the activity in question. For example, cm line 6 in Figure 1, the AI .I_data_take activity must start after the end of the SAD_ changer activity by 100-300 seconds. This constraint tells the scheduler that the SAD_changer activity mast be completed at least 100 seconds before the AI .I_data_take activity starts. Using the same method, the start or end time of any activity can be specified relative to the start or end time of the owner activity. If the time duration is specified as [0,0], the start or end times will coincide exactly. The "contains" constraint is used for activities that fall within the owner activity. This constraint definition combines a starts_ before start of and an ends_after end of pair of constraints. For example, line 8 defines a constraint for the SAD_ user activity that is contained within owner activity Al .I_data_ take. The first two and last two numbers in the constraint represent ranges of time, which separate the start times between the two activities and the cad times between the two activities. SAD_user must start exactly four seconds (4,4) after- the start of ALI_data_take but the cad time can coincide with the cad time of Al .I_data_take or Up to 10 seconds (O, 10) earlier. This relationship is graphically represented in Figure 2. Because the ALI_data_take activity has a variable duration, ASPEN automatically chooses a duration that satisfies the above temporal constraints.

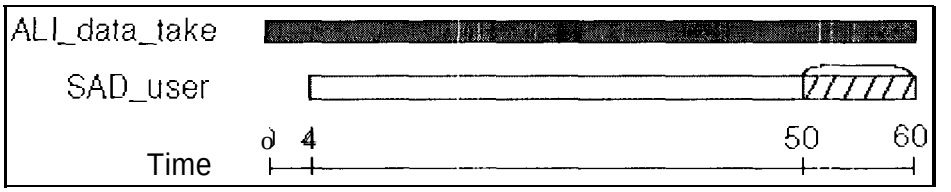

Figure 2: Constraint Relationship: contains

The [contained_by] constraint is used for activities that are the same size or larger thin the owner activity. For example, line 9 defines a constraint for activity SAD_user that starts exactly 300 seconds before the start of and ends 1-16 seconds after the end of activity Al .I_data_ take.

Subactivities arc activities that can be scheduled any time within the parent activity subject to resource constraints within the subactivity. Subactivities are similar to the constraint-defined activities without the exact temporal relationship between the parent and subactivities. For example, line 10 defines subactivities ALI_user_data and Al .I_dark_count. These activities must fall within the temporal range of the parent activity AI. I_data_take. The main difference between "constraint" and "subactivities" is that "constraints" can be satisfied by any activity in the schedule. Subactivities arc always created when defined within a parent activity. "1'here is a one to one relationship from parents to subactivities.

Reservations are used to reserve a portion of a resource or state for the duration of the activity. There arc two types of resource reservations: atomic and aggregate. Line 11 of Figure 1 contains examples of an atomic reservation (processor) and aggregate reservation (array_power). The processor reservation reserves the processor for the duration of the activity. No other activities can use the processor during this time. The array_ power reservation uses 80 units of array_power for the duration of the activity. If the array_ power *were* a depletable resource, the 80 units would be reserved from the start of the activity until the end of the planning horizon.

State reservation either change the state of a state variable or reserve a state for the duration of an activity. Line 16 of Figure 1 changes the state of the solar_array state variable to the value of parameter sad]. Line 17 of Figure I reserves the "open" state of the aperture state variable for the duration of the activity. If the aperture state variable was in a state other than "open" prior to this activity, the scheduler would have to create a state changer activity to change the state to "open. "

Resources

Resources arc a description of a profile of a physical resource over time. '1'here arc four types of resources: atomic, concurrency, depletable, and non-depletable. Atomic resources are physical devices that can only be used (reserved) by one activity at a time. Examples of atomic resources include: science instrument, star tracker, reaction wheel, or CPU. Concurrency resources arc similar to atomic except they must be made available to the activity before they arc reserved. An example would be a telecommunications downlink pass. The telecommunications station would have to be made available before the spacecraft could initiate a downlink. Non-depletable resources are resources that can used by more than one activity at a time and do not need to be replenished. Each activity can use a different quantity of the resource. Examples include solar array power and the 1773 bus. Depletable resources arc similar to non-depletable except that their capacity is diminished after use. In some cases their capacity can be replenished (bat(cry energy, memory capacity) and in other cases it cannot (fuel). A summary of the four types of resources is presented in Table 1.

Resource Type	Properties
Atomic	Always available when not in USC, only 1 user at a time Ex: science instrument, star tracker, reaction wheel, dedicated CPU
Concurrency	Only available when made available, only 1 user at a time Ex: telecommunicat ions downlink pass
Norl-depletable	Always available when not in use, many users can use different quantities Ex: solar array power and 1773 bus
Depletable	Capacity is diminished after USC, may or may not be replenished by another activity Ex: battery energy, memory capacity, fuel

"1'able 1: Resource Types

The four types of resources are defined in lines 1,6, 12, and 18 of Figure 3. The definition includes the name followed by a pair of braces and a semi-colon similar to the C language syntax. The type is onc of: atomic, concurrency, depletable and non-depletable. The name and type arc the only required components of a resource definition. Once the resource is defined, it can be instant iated in the initial slate file. Generally, this instance will consist of just the resource name followed by the instantiated name and a pair of braces. Note: concurrency resources are not yet implemented.

The capacity of a resource can be specified as a constant, list or range. A range would be used if several similar resources

with specific capacities were defined when the resources were instantiated.

```
1   Resource ALI{
2     Type = atomic;
3     Capacity = 1;
4   };
5
6   Resource Solar_array {
7     Type = non_depletable;
8     Capacity = 600; // watts
9   };
10
11
12  Resource warp_storage{
13    Type = depletable;
14    Capacity = 40000;  // Mbits
15   };
16
17
18  Resource Propellant {
19    Type = depletable;
20    Capacity = 15;  // kg
21   };
```

Figure 3: Resource Examples

An atomic resource has a unit capacity and does not have to be explicitly set such as on line-3 of Figure 3. Depletable and non-depletable resources definitions can contain a minimum capacity such as in lines **9, 15,** and **21** of Figure 3.

States

A device, subsystem, or system may be represented by a state variable that gives information about its state over time. The state variable contains the state profile, which is defined as an enumerated type. Some examples of possible states arc: on, off, open, closed, record, playback, standby or idle. States can bc reserved or changed by activities. A state variable must equal some state at every time. At the beginning of a planning horizon, this slate is just the default state. Figure **4** contains two examples of state variable definitions.

A state variable is defined in lines 1 and 7 of Figure 4. The definition includes the name followed by a pair of braces and a semi-colon similar to the 'C' language syntax. Lines **2** and 8 contain a list of the states the state variable can contain. The default state must be defined and must be one of the states in the list. Once the stale variable is defined, it can bc instantiated in the initial state file.

The allowable state transitions between stales can be indicated using the 'transitions' keyword with a forward(->) arrow, a bi-directional arrow (<->), or with the 'al]' keyword (e.g., all<->all).

Parameters

The ASPEN modeling language includes parameters, which are variables or constants. Parameters can consist of integers, strings, floating points, booleans, or lists or ranges of any of these. Parameters can bc defined as enumerated types for a list of slates in a stale variable. Ranges of values can also be used. Some examples arc:

- parameter string ALI_ mode { domain = ("data'>, "idle", "standby", "off");) ;
- parameter int warprange { domain= [1 **,40960];);**

In the first example, the Al.]_ mode parameter can take on any of the four values in the list. In the second example, the warprange parameter can be any integer in the indicated range from I 1040960.

EO-1 MODEL

EO-1 is an Earth imaging satellite that is part of the New Millennium Program of technology validation missions. The NASA Goddard Space Flight Center is responsible for project management. The purpose of EO- 1 is to validate new technologies that can be used on future Landsat class Earth remote sensing missions. In fact, EO-1 will be flying in formation onc minute behind Landsat-7, with the goal of imaging as many of the same targets as possible. EO- 1 will be using the Landsat 7 daily scene list as an input file of potential EO-1 targets.

The main activity in EO-1 operations is the Advanced Land Imager (A1.1) data take. The AI.1 instrument contains six separate detectors that output data simultaneously. Onc image takes a total of 24 seconds and consumes about 19 gigabits of data on the sol id state recorder (WARP). Because the capacity of the WARP is only 40 Gbits, it is important to plan the data takes and downlinks to maximize the amount of data returned. Due to limited amount o f downlink time available, only four data takes pcr day can be taken. Data takes can bc prioritized based on the following parameters:

- Cloud cover over the region to be imaged
- Sun angle at the region to be imaged
- Ability to return the data before overflowing the WARP recorder
- Images coinciding with Landsat 7 images
- Imaging of scientifically interesting areas

Each EO - 1 data take has several conditions that must bc satisfied before and after the data take occurs. These conditions are listed below:

Before:

- Change the ACS mode to science
- Change the solar array to a fixed orientation
- Open the All aperture
- Change the data rate to high rate mode

After:

- Close the ALI aperture
- Take one second of calibration data
- Change the ACS, solar awry, and data rate modes brick to the previous values

Each of these conditions is modeled as temporal constraints in the ALI data take activity. The data take activity itself is only a 24-second activity. The constraints on the activity

```
1   State_variable ALI_sv (
2         states = ( "data", "standby", "idle","off" ) ;
3         transitions ( "standby"->"data","data"->"standby", "idle" ->"standby",
                                "standby"->"idle", "off"->"idle,', "idle->off");
4         default_state = "idle";
5   };
6
7   State_variable aperture_sv {
8         states = ( "open", "closed");
9         transitions = ( "open"->"closed","closed"->"open" );
10        default_state = "closed";
11  }};
```

Figure 4: State Variable Examples

span period of five minutes before and one minute after the bounds of the activity. The constraints on the activity could have been modeled as subactivities. The reason we chose to mode] these activities as constraints is because of their tight temporal constraints. l'hc data take activity tweaks down into 14 separate activities as listed in Table 2.

ALI Scene Collection	
ALI_data_take	aperture_changer
ALI_user_data	engdata_user
ALI_user_standby	engdata_changer
ALI_changer	ACS_user
SAD_user	ACS_changer
SAD_changer	cloud_cover_changer
aperture_user	sun_angle_changer

Table 2: EO-1 Science Activities

The ALI must be calibrated by viewing the sun or the moon regularly. The sun calibration involves pointing at the sun and changing the aperture filter several limes. The moon calibration points at the lunar limb and pans across the moon using each of the detectors. Similar to the data take activities, the calibrations involve several constraints, The calibration activities and constraints are listed in'1'able 3.

EO-1 communicatio n activities are modeled as follows:

1. An input file gives the times at which the ground station is in view 01 the satellite.
2. The in view times are converted into a state variable with the value 'invicw' or 'outview.'
3. The planner chooses communication links during these in view times.
4. The communicatio n link is broken down into uplinks (if required) and downlinks.

ALI_sun_calibration
slew_to_sun
aper_test_changer
ALI_moon_ calibration
moon__cal_ms_ pan
slew_ to_ moon
ramp_up_pitch_slew
ramp_down_pitch_slew
roll_ to_next_position

Table 3: EO-1 Calibration Activi es

The EO-1 model also includes initialization activities for power, propellant, and memory. These activities arc used to keep track of consumable resources from the previous planning period.

A key-word 'command' is used for activities that represent an EO-1 spacecraft command, When the command keyword is included in the activity definition, along with the command name, the spacecraft command output file will include a time tagged command for that activity.
The EO-1 spacecraft resources are modeled as either depletable or non-depletable. It was not necessary to model every physical device on EO-1 because many devices consumed a constant power and did not interact with any spacecraft activities. The power of these devices is included in the power_init activity. The resources that arc modeled are listed in "1'able 4.

Non-Depletable		Depletable
ALI	Processor	Battery
S_band_Receiver	Bus_1773	Warp_storage
Transponders	Cat_bed_heater	Propellant
solar. array	WFF	
ACDSE	DSN	
Warp		

Table 4: EO-1 Resources

The EO-1 ASPEN model has ten different state variables which are listed in Table 5. Most of these state variables arc used to represent the state of a spacecraft resource. The states arc used in activities that require a resource to be in a particular state. These requirements are specified in the reservations of the activity. For example, the EO-1 data take activity requires the WARP state variable to be in record mode during the period of imaging. This requirement ensures that the data is being recorded during the imaging operation. Activities arc defined that either change or use a particular state of a state variable. These acti vities usually contain a command keyword that corresponds to an EO-1 spacecraft command.

Creating the EO-1 Model

The modeling language has been designed such that it can model a physical spacecraft system directly. It is a descriptive language that allows an engineer to directly represent the physical spacecraft information in the model. in fact, the EO-1 model was created by an engineer (first author, Rob Sherwood) who had no knowledge of the

software or its algorithms and procedures. He successfully created the model by simply taking the EO-I spacecraft information and pulling it into the modeling language syntax. This process took three weeks, Another similar model for the Spacecraft Interferometr y Mission took less than two clays.

State Variables	
Variable	*Possible States*
ALI_sv	data, standby, idle, off
SAD_sv	off, tracking, fixed
aperture_sv	open, closed
aperture_test_sv	small, med, large, blank
engdata_sv	high, low
ACS_sv	nadir, low_jitter, standby, safe, orbit_adjust,
WARP_sv	off, idle, reed, playback
Cloud_Cover_sv	low, med_low, med, med_high, high, none
Sun_Angle_sv	low, med, high, none
WFF_inview_sv	in, out

Table 5: EO-1 State Variables

The modeling language is flexible and allows for different ways of representing the same information. Therefore, there is no onc correct model for a given spacecraft. The EO-1 model is constrained to have certain state variables, for example, as determined by the mission, but, on the other hand, could have different ways of representing constrains between activities.

END-TO-END PLANNING SYSTEM

The goal of this EO- 1 work is to produce an automated on-board planning system for spacecraft commanding of the EO-1 satellite. The system will bc validated after launch on the ground. As a ground based planner, the inputs to ASPEN include:

- Landsat-7 cloud cover and sun angle predictions
- Current power-, propellant, and memory levels
- Sun, moon, and sky calibration requests
- Ground station view files
- Mancuver requests

Once ASPEN is delivered to the EO-1 project, there will only bc minor changes made to the model to integrate ASPEN into the existing operations. Wc plan to automate the loading of the input files such as cloud cover and sun angle predictions into ASPEN, and link the output schedule of ASPEN directly to the existing EO-I software. In fact, the creation of the input files can bc invoked from external calls from the ASPEN GUI. With ASPEN linked directly to its input files through the GUI, the 110-1 planning process will be seamless and efficient.

The output of the ground based validation of the planner will bc a text list of time tagged commands that will bc translated into binary spacecraft commands by the ground system load generation utility. This utility is already built into the EO-1 ground system.

The on-board planning system will require upload of the ground station view files and maneuver requests. The cloud cover could be obtained by using the ALI science instrument to examine the CLOUDIs before a scene. After the image is taken, the cloud data would be analyzed to determine if the scene should be saved and downlinked. Clouded scenes would be erased from the WARP and a new scene would be planned to take its place.

EO-1 Model in Action

Generating EO- I mission operations schedules is a fast process. Given a set of EO-1 requests, ASPEN will generate a conflicl-free schedule within the order of a few minutes for lengthy schedules, and within seconds for simpler schedules. For example, for 162110-1 activities, it takes ASPEN 3.53 seconds (on a SUN Ultra-2) to produce a conflict-free schedule. '1'here are no EO-1 schedules that take more than a minute to schedule, but with other spacecraft models with more activities and lengthier schedules, wc have seen a maximum of five minutes to produce a corrffict-free schedule.

In addition to having the activity requests specified in advance, the user can make changes to the schedule from the GUI as needed. For example, the user could add an ALI_data_take activity. If this caused conflicts in the schedule, then ASPEN would resolve the conflicts. This whole process takes seconds to execute. For example, with the EO-1 model, if wc add three ALI_data_ take activities in the GUI (randomly placed), this causes 34 conflicts. Resolving all conflicts, and producing a corrflict-free schedule takes 1.54 seconds (on a SUN Ultra-2). This means that it is solving approximately 17 conflicts per second. (Adding just onc data-take activity causes a large number of con flicts because of the constraints between activities and the states required by different activi ties.)
Currently, activities can be given a particular score, and high-level preferences (such as resource max usage) can be indicated which also determine scores for activities. The generated schedule is then given a score based on the activities' scores. Using this score, the user can then choose one generated schedule over another. Wc arc presently working on an algorithm that will automatically optimize schedules.

Limitations of ASPEN

The algorithms and data structures used in ASPEN impose some limits on what ASPEN can model and solve. For example, the iterative repair algorithm used to repair schedules and make them conflict-free is a local search algorithm and therefore cannot solve problems where local search techniques do not work. Wc are currently developing a search algorithm framework which will allow many types of search strategies (global and local) to be used in ASPEN. In addition to the current local search algorithm constraint, at present ASPEN also presumes that the units in tbc

timelines, which can take linear or exponential functions, are constant value over a unit. In the future, ASPEN may have units whose value varies over the unit, but now it is a known limitation. Lastly, although the modeling language is expressive, it is limited to what can be defined within the existing modeling language syntax. For example, it is currently difficult to model the power interaction between the solar array and batteries. When the EO-1 satellite is occulted by the Earth, activities in the plan which use solar array power must instead use battery power. The reverse is true when EO-1 is in direct sunlight. There are also periods where both solar array and batteries are used for power due to partial illumination of the solar array. Combining these effects with the complex charging and discharging cycles of the batteries creates a difficult problem to model. We are currently improving the modeling language so complex interactions such as these can be successfully modeled.

CONCLUSIONS

Modeling EO-1 mission operations in ASPEN is easy and compact. The entire EO-1 model consisting of the activities, parameters, reservations, resources, and state variables as described above, is represented in approximately 700 lines (in plain text files) which also includes comments and headers. The simplicity of the modeling language will allow the operations personnel to easily change the model when needed. The changes will not require a recompile of the code.

We have successfully modeled EO-1 mission operation activities with ASPEN. We have created a model which encapsulates information about: data takes, calibration activities, maneuvers, uplinks, downlinks, validation activities, cloud cover and sun angle states, and initialization activities of power, propellant, and data storage. Using this model with ASPEN will enable EO-1 to function with a very *small* operations team.

REFERENCES

[1] J. Allen, J. Hendler, and A. Tate, *Readings in Planning*, Morgan Kaufmann, 1994.

[2] S. Chien, D. Decoste, R. Doyle, and P. Stolorz, "Making an Impact: Artificial Intelligence at the Jet Propulsion Laboratory," *AI Magazine*, 18(1), *103-122*, 1997.

[3] A. Fukunaga, G. Rabideau, S. Chien, and D. Yan, "ASPEN: A Framework for Automated Planning and Scheduling of Spacecraft Control and Operations," *Proceedings of the International Symposium on AI, Robotics and Automation in Space (i-SAIRAS)*, Tokyo, Japan, 1997.

[4] S.F. Smith, O. Lassila, and M. Becker, "Configurable Mixed-Initiative Systems for Planning and Scheduling," *Advanced Planning Technology*, AAAI Press, 1996.

[5] D. Speer, P. Hestness, M. Perry, and B. Stabnow, The New Millennium Program EO-1 Mission and Spacecraft Design Concept, In Proceedings of the IEEE Aerospace Conference, v. 4, pp. 207-227, Snowmass, CO, 1997.

[6] M. Zweben and M. Fox, *Intelligent Scheduling*, Morgan Kaufmann, 1994.

BIBLIOGRAPHY

Rob Sherwood *is a Member of Technical Staff at the Jet Propulsion Laboratory, California Institute of Technology. He holds a B.S. in Aerospace Engineering from University of Colorado at Boulder, and a M.S. in Mechanical Engineering from the University of California at Los Angeles. He is currently pursuing an M.B.A. at Loyola-Marymount University. Robert has received 4 NASA Achievement Awards for his work in Spacecraft Mission Operations. He is currently working on several projects involving Planning and Scheduling technologies.*

Anita Govindjee *is a Member of Technical Staff in the Artificial Intelligence Group at the Jet Propulsion Laboratory, California Institute of Technology. She holds a M.S. in Computer Science from Stanford University and a B.S. in Computer Science from the University of Illinois. Her research interests are in artificial intelligence and cognitive science.*

David S. Yan *is a Member of the Technical Staff in the Artificial Intelligence Group at the Jet Propulsion Laboratory, California Institute of Technology. He holds a B.S.E.T. in Electrical Engineering and Computer Science from the University of California at Berkeley. He is pursuing his M.S. degree in Computer Science at Stanford University. His research interests include automated planning/scheduling, operating systems, computer architecture and computer networks.*

Gregg Rabideau *is a Member of the Technical Staff in the Artificial Intelligence Group at the Jet Propulsion Laboratory, California Institute of Technology. His main focus is in research and development of planning and scheduling systems for automated spacecraft commanding. Projects include planning and scheduling for the first deep-space mission of NASA's New Millennium Program, and for*

design trades analysis for the Pluto Express project. Gregg holds both a B.S. and M.S. degree in Computer Science from the University of Illinois where he specialized in Artificial Intelligence.

***Steve Chien** is Technical Group Supervisor of the Artificial Intelligence Group of the Jet Propulsion Laboratory, California Institute of Technology where he leads efforts in research and development of automated planning and scheduling systems. He is also an adjunct assistant professor in the Department of Computer Science at the University of Southern California. He holds a B.S., M.S., and Ph.D. in Computer Science from the University of Illinois. His research interests are in the areas of: planning and scheduling, operations research, and machine learning.*

***Alex** S. **Fukunaga** is a Member of the Technical Staff in the Artificial Intelligence Group at the Jet Propulsion Laboratory, California Institute of Technology. He holds an A.B. in Computer Science from Harvard University, and a M.S. in Computer Science from the University of California at Los Angeles, where he is currently a Ph.D. student. His research interests include optimization, decision theory, search, machine learning, and automated planning/scheduling.*

ACKNOWLEDGMENTS

This paper describes work performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.