ASPEN: A Framework for Automated Planning and Scheduling of Spacecraft Control and Operations

Alex S. Fukunaga Gregg Rabideau Steve Chien David Yan

Jet Propulsion Laboratory
California Institute of Technology*
4800 Oak Grove Drive, MS 525-3660
Pasadena, CA 91109-8099, USA
{alex.fukunaga,gregg.rabideau,steve.chien,david.yan}@jpl.nasa.gov

Abstract

A number of successful applications of automated planning and scheduling applications to spacecraft operations have recently been reported in the literature. However, these applications have been one-of-a-kind applications that required a substantial amount of development effort. In this paper, we describe ASPEN (Automated Planning/Scheduling Environment), a modular, reconfigurable application framework which is capable of supporting a wide variety of planning and scheduling applications. We describe the architecture of ASPEN, as well as a number of current spacecraft control/operations applications in progress.

1 INTRODUCTION

Automated planning/scheduling technologies have great promise in reducing operations cost and increasing the autonomy of aerospace systems. Planning[1] is the selection and sequencing of activities such that they achieve one or more goals and satisfy a set of domain constraints. Scheduling selects among alternative plans and assigns resources and times for each activity so that the assignments obey the temporal restrictions between activities and the capacity limitations of a set of shared resources. In addition, scheduling is an optimization task in which metrics such as tardiness and makespan are minimized. Scheduling is a classical combinatorial problem that has long been studied by researchers in operations research. While traditional operations research approaches (c.f. [7]) have focused on optimal solutions for highly restricted classes of problems, there has been much recent interest in the heuristic, constraint-based approaches that are applicable to practical domains.

Traditionally, the problems of planning and scheduling have been studied separately. Recently, approaches that integrate both the planning and scheduling process together under a unifying framework in which plans are generated and scheduled simultaneously by a single system (as opposed to using separate planning and scheduling systems) have been developed. Some recent aerospace applications of hybrid planner/schedulers include [10, 11, 15].

Although the benefits of applying planning/scheduling technology can be significant, developing real-world, planning/scheduling systems is often an extremely time-consuming task. Modeling a complex domain requires an expressive modeling language, as well as data structures that represent the constraints expressed in the domain model. In addition, complex data structures and algorithms that support incremental modifications to candidate plans/schedules need to be designed and implemented.

In order to enable the rapid development of automated scheduling systems for NASA applications, we have developed ASPEN (Automated Scheduling and Planning ENvironment), a reusable, configurable, generic planning/scheduling application framework. An application framework [14] is a class library (i.e., a reusable set of software components) that provides the functionality of the components found in prototypical instances of a particular application domain. Frameworks anticipate much of an application's design, which is reused in all applications based on the framework. This implies a significant reduction in the amount of code necessary to implement successive systems.

The reusable components provided by ASPEN include:

- An expressive constraint modeling language to allow the user to naturally define the application domain;
- A constraint management system for representing and maintaining spacecraft operability and resource constraints, as well as activity requirements;

*In Proc. Int. Symposium on AI, Robotics and Automation in Space (i-SAIRAS), Tokyo, Japan, 1997.

[1] We take these definitions of planning/scheduling from [6].

- A temporal reasoning system for expressing and maintaining temporal constraints; and
- A graphical interface for visualizing plans/schedules (for use in mixed-initiative systems in which the problem solving process is interactive).

ASPEN is currently being utilized in the development of an automated planner/scheduler for commanding the New Millennium EO-1 satellite and a naval communications satellite, as well as a scheduler for the ground maintenance for the Reusable Launch Vehicle and a design analysis tool for the Pluto Express spacecraft. The rest of the paper is organized as follows: Section 2 describes the architecture of ASPEN and its components. Section 3 describes some current applications of the ASPEN framework, including ground maintenance scheduling for the Reusable Launch Vehicle, as well as operations planning/scheduling for two autonomous satellites. Finally, Section 4 describes related work.

2 The ASPEN Architecture

The development of an application framework for a particular domain implies a standardized approach to implementing systems for that domain, and a commitment by the framework developer to support applications that conform to that standardized approach. It is impractical to develop a framework to support all viable approaches to planning and scheduling. Numerous, widely divergent approaches to planning and scheduling have been developed (cf. [3, 20]. Since planning/scheduling are currently very active areas of research, there is no clear consensus on which approaches are most useful. Thus, we restricted the scope of our framework to approaches that had been found useful for NASA applications in the past.[2]

By analyzing our previous experience with building planning/scheduling systems, (cf. [15, 11]), as well as requirements for current and future applications, we abstracted a set of requirements that is flexible enough to support a wide range of applications, and developed the components described below:

2.1 Activity Database

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, end time, and a duration. Activities can use one or more resources. All activities in a plan/schedule are elements of the Activity Database (ADB), which maintains the state of all of the activities in the current plan/schedule, and serves as the integrating component that provides an interface to all of the other classes.

[2]See [4] for an overview of planning/scheduling applications recently developed at JPL.

One function of the ADB is to represent and maintain hierarchical relationships between activities. Activities can contain other activities as subactivities; this facility can be used to reason about the plan/schedule at various levels of abstractions (e.g., a scheduler can first reason about a set of activities without considering that each of those activities are themselves composed of a set of subactivities – this can make various reasoning tasks much more computationally tractable.

Temporal and resource-constraints between activities are also represented in the ADB. Although most of the actual computational mechanisms that maintain these constraints are implemented in other the modules described below, the protocol that a search algorithm uses to access constraints in the context of a plan/schedule is implemented in the ADB. For example, although the resource timelines are responsible for detecting overuse of resources by activities, the ADB maintains data structures that indicate the assignment of activities to specific timelines, so that one can efficiently ask queries such as, "which resources does this activity use?"

As another example: although the temporal constraint network (see below) is responsible for maintaining temporal constraints between individual activities, the ADB is responsible for global constraints. (e.g., the ADB contains global constraints such as: "all activities occur after the start of the scheduling horizon." When an activity is created, the temporal constraint that the activity occurs after the horizon is created automatically by the ADB).

2.2 Temporal Constraint Network

A Temporal Constraint Network (TCN) is a graph data structure that represents temporal constraints between activities. A temporal constraint describes the temporal relationship between an activity and other activities and/or the scheduling horizon, and impose an ordering on the set of activities. The TCN implements a Simple Temporal Problem, as defined in [5], and represents a set of constraints, all of which must be satisfied at any given time, i.e., it represents the conjunct of all active constraints between activities in the ADB. Activities are represented in the TCN as pairs of time points, where each time point corresponds to the beginning or end of an activity, and the edges in the TCN graph represent the constraints on the temporal distance between the time points. The TCN can be queried as to whether the temporal constraints currently imposed between the activities are consistent.

2.3 Resource Timelines

Resource timelines are used to reason about the usage of physical resources by activities. Capacity conflicts are detected if the aggregate usage of a resource exceeds its capacity at any given time. Several subclasses of resource timelines are implemented, including depletable resource timelines used to model consumable resources (e.g., fuel), and non-depletable resources that are used to model resources which are not actually consumed by usage, but are instead "reserved" for a period of time (e.g., a piece of equipment). Our current model of resource usage is discrete. That is, if we specify that an activity such as move-forward uses 2 units of fuel, then both of these units are modeled as being immediately consumed at the beginning of the activity. This is a discrete approximation, since the usage of the fuel may be better modeled as a linear function such as $usage(t) = 2t/(activitylength)$, where t is the time elapsed since the beginning of the activity, and $activitylength$ is the duration of the activity.

2.4 State Timelines

State timelines represent arbitrary attributes, or states, that can change over time. Each state can have several possible values; at any given time, a state has exactly one of these values. Activities can either change or use states. For example, a door-open activity would set the state of door to be open, while an enter-building activity would require that the state of door be open. As activities are placed/moved in time, the state timeline updates the values of the state, and detects possible inconsistencies or conflicts that can be introduced as a result. For example, an activity that requires that the door be open is placed at time t, then the state timeline checks to verify that the door is in fact open at time t. Otherwise, a state constraint violation is indicated. Users can define legal sequences of state transitions. The state timeline class will detect illegal transition sequences if they are introduced into the timeline.

For example, consider modeling a traffic light with a state timeline, *traffic-light*. The possible values are $green$, $yellow$, and red. The legal state value transitions are: $green$ to $yellow$, $yellow$ to red, red to $green$. All other transitions are illegal.

2.5 Parameter Dependency Network

Each activity has a number of parameters that are either user-defined or computed by the system, such as start time, end time, duration, any resources it uses, any states it changes/uses, etc. In ASPEN, it is possible to create dependencies between pairs of parameters within the same activity, or between pairs of parameters defined in different parameters. A dependency between two parameters p_1 and p_2 is defined as a function from one parameter to another, $p_1 = f(p_2)$, where $f(x)$ is an arbitrary function whose input is the same type as p2, and whose output has the type of p1. These dependencies are represented and maintained in a Parameter Dependency Network (PDN). The PDN maintains all dependencies between parameters, so that at any given time, all dependency relations are satisfied. Note that if there exists a dependency such that $p_1 = f(p2)$, its inverse dependency, $p_2 = f^{-1}(p_1)$ does not necessary exist, unless the user specifies the inverse relationship and specifies the inverse dependency as well.

Note that the TCN can be seen as a special case of a PDN in which the functional relationships between the parameters (activity start/end times and durations) is a distance relationship, and for which very efficient constraint propagation algorithms have been implemented.

In general, as commonly used special cases of functional dependencies between parameters (such as temporal distance relationships), it can be useful to develop special dependency networks that implement efficient constraint propagation algorithms that take advantage of the special structure of these dependencies, instead of relying on the general mechanism offered by the PDN.

Such special-purpose dependency networks can be implemented as subclasses of the abstract parameter dependency network, or (if the protocol that must be supported is sufficiently unique) abstracted out as a separate basic component of ASPEN, as was done with the TCN.

2.6 Planning/Scheduling Algorithms

The search algorithm in a planning/scheduling system searches for a valid, possibly near-optimal plan/schedule. The ASPEN framework has the flexibility to support a wide range of scheduling algorithms, including the two major classes of AI scheduling algorithms: constructive and repair-based algorithms.

Constructive algorithms (e.g., [6]) incrementally construct a valid schedule, ensuring that at every step, the partial schedule constructed so far is valid. When a complete schedule is constructed, it is therefore guaranteed to be valid. Repair-based algorithms(cf. [9, 19]) generate a possibly invalid complete schedule using either random or greedy techniques. Then, at every iteration, the scheduled is analyzed, and repair heuristics that attempt to eliminate conflicts in the schedule are iteratively applied until a valid schedule is found.

The search algorithms that have currently been implemented include:

- forward dispatch, a greedy, constructive algorithm;
- IRS, a constructive, backtracking algorithm based on [11]; and
- DCAPS, a iterative repair based algorithm based on [15].

2.7 Graphical User Interface

The ASPEN Graphical User Interface (GUI) component provides tools for graphically displaying and manipulating schedules. Resource and state timelines are displayed. Activities are overlayed on the timelines, and users can directly manipulate activities using standard drag-and-drop procedures.

```
Activity prevalve_removal {
  duration = 15
  slot subsystem
  after prevalve_prep with (subsystem == this.subsystem)
  before prevalve_replace with (subsystem == this.subsystem)
  Reservation hydraulic_lift_usage {
    resource = hydraulic_lift;
    usage = 1;
    duration = 5;
  }
  requires state prevalve-purged TRUE
  requires state prevalve-illuminated TRUE
}
Resource hydraulic_lift {
  type non-depletable
  quantity 1
}
```

Figure 1: Sample of ASPEN modeling language (part of the Reusable Launch Vehicle maintenance model). This describes an activity for removing the prevalve of an engine subsystem.

2.8 Extending ASPEN for Applications

There are two means by which ASPEN can be extended and specialized for a particular application. These are:

- Creation of domain-specific models using the modeling language, and
- Extension of the application framework code.

The modeling language is used to specify domain-specific constraints and activities. Figure 1 shows part of a domain model specified in the ASPEN modeling language.

The base ASPEN framework, including the modeling language is sufficiently extensible to support a range of applications without any extensions to the code of the framework itself (e.g., the Reusable Launch Vehicle ground maintenance scheduling application is directly derived from the framework by simply specifying a model file).

Extensions to the framework code need to be made when changes in the behavior of ASPEN components are required. This includes two classes of extensions: epistemological and heuristic.[3] Epistemological extensions are necessary when new representational capabilities are in order to model a new domain. For example, if we wanted to implement a new type of resource timeline which had a more sophisticated, continuous model of resource usage[4], then a new subclass of the resource timeline abstract class would need to be implemented. Heuristic extensions customize the behavior of the framework to improve the quality of solutions found or the time to find good solutions for a particular domain.[5] Examples of heuristic extensions include new repair heuristics for a repair-based scheduler, or an entirely new search algorithm.

3 APPLICATIONS OF ASPEN

In this section, we describe ongoing applications of the ASPEN scheduling system to: generation of spacecraft command sequences for the New Millennium Earth Observing One satellite and the U.S. Navy UHF Follow On One (UFO-1) satellite; generation of mission operations sequences to assist in design analysis for science and operability; and rapid generation of plans for maintenance and refurbishing for Highly Reusable Space Transportation.

3.1 Spacecraft Commanding

The primary application area for the ASPEN scheduling system is generation of spacecraft command sequences from high level goal specifications.

In this role, automated scheduling systems will encoding of complex 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 use of planning and scheduling technology.

By automating this process and encapsulating the operation specific knowledge we hope to allow spacecraft commanding by non-operations personnel, hence allowing significant reductions in mission operations workforce with the eventual goal of allowing direct user commanding (e.g., commanding by scientists).

Current ASPEN applications to spacecraft commanding focus on two missions: the New Millennium Earth Observing One (NM EO-1) satellite (to be launched in late 1998) and the U.S. Navy UHF Follow On One (UFO-1) satellite (currently in orbit). NM EO-1 [18] is a earth imaging satellite featuring an advanced multi-spectral imaging device. For this mission, operations consists of managing spacecraft operability constraints (power, thermal, pointing, buffers, consumables, engineering downlinks, etc.) and science goals (imaging of specific targets within particular observation parameters). Of particular difficulty is managing the downlinks as the amount of data generated by the imaging device is quite large and ground contacts are a limited resource.

The current ASPEN EO-1 scheduler generates an initial schedule using forward sweeping greedy dispatch to generate an initial schedule, then uses the DCAPS iterative repair algorithm to resolve state, resource, and temporal conflicts.

[3] This classification follows [8].

[4] Recall from Section 3 that we use a simple, discrete resource usage model.

[5] This implies that the framework is, in principle capable of eventually finding some solution without a heuristic extensions.

Another ongoing effort in the area of spacecraft commanding is the development of an advanced commanding system for the U.S. Navy UFO-1 satellite [1]. UFO-1 is an on-orbit testbed managed by the U.S. Naval Academy Space Artificial Intelligence Lab (SAIL) at Annapolis. In this collaboration, SAIL is developing an uplink, downlink, basic data transport, and commanding capability to be interfaced with an advanced planning and scheduling engine (ASPEN). In this application, ASPEN will allow high level commanding of the UFO-1 satellite to perform high level functions such as: auto pitch momentum dumping, preparation for eclipse season, delta-V maneuvers, IRU warmup and turnon, battery cell pressure bias calibration, delta inclination maneuvers, and other engineering housekeeping functions. The ASPEN scheduling engine then performs appropriate expansion and conflict resolution to generate lower level command sequences to achieve the higher level goals.

3.2 Design Evaluation

ASPEN is also being applied in the Pluto Express [2] for the dual purposes of science planning and design evaluation for science and operability [2]. In support of science planning, we are developing high level models of proposed PX spacecraft to assist in automated generation of science data acquisition plans (e.g., high level activity sequences) from high level science goals to assist in developing science plans for mission profiling.

This same capability to generate science plans is being used to evaluate candidate spacecraft designs from the standpoint of emergent design aspects such as science return and operability. This spin-off application arises from the observation that often it is difficult to determine how well a given spacecraft design will perform without fleshing out approximate operations sequences for critical phases of the mission (e.g., encounter). In order to address this difficulty, we are developing a design analysis tool which accepts as input: a candidate spacecraft design (and operations constraints, models, etc.); a set of engineering and science objectives; and a set of scoring functions to assess how well a sequence achieves the objectives. This tool then applies an ASPEN-based planner/scheduler to generate a candidate sequence; then uses the scoring function to score this sequence in terms of the aspects of science, operability, etc. This enables design teams to rapidly and impartially evaluate large numbers of spacecraft designs with little effort, thus allowing improved analysis of design tradeoffs to enhance science and operations concerns for future missions.

3.3 Maintenance Scheduling

As part of the NASA Highly Reusable Space Transportation (HRST) program[6] we have been developing and demonstrating advanced scheduling systems for the rapid generation and revision of plans for maintenance and refurbishment of highly reusable launch vehicles. In this application, real-time telemetry downlinked either during flight or immediately after flight would be analyzed to automatically generate a set of maintenance requests, which would then be transformed into a refurbishment plan by an automated planning and scheduling system which would account for available equipment and resources as well as the intricacies of the refurbishment procedures of the highly complex propulsion systems. The end target is to allow a turnaround of several hours for the HRST spacecraft to support a flight frequency on the order of several flight per day.[7] If the maintenance schedule can be generated using in-flight telemetry then the refurbishment process can be speeded even further by allowing for downlinking of requests for pre-positioning of equipment and resources to minimize schedule delay.

Once the actual maintenance plan has been generated, the planning tool continues to be of use in two ways. First, in many cases there can be several mutually exclusive maintenance activities which can be performed next. Via lookahead and critical path analysis automated scheduling software can determine the next activities to enable the minimal makespan (overall schedule execution time). Second, as unexpected events arise (such as equipment failures, resource unavailabilities, and schedule slippage), the automated scheduling software has the ability to revise the schedule so as to minimize schedule disruption (movement of activities and resources from their original assignments) and schedule slippage (delay of the completion of the overall refurbishment).

In order to test and validate this technology we have been utilizing test maintenance procedures Specifically, we used the maintenance procedures developed for the LO_2 and LH_2 propulsion systems for the Rockwell International X-33 Reusable Launch Vehicle.[8] The procedures derived for maintaining and refurbishing the test articles provided a rich testbed for Space Propulsion System Maintenance Scheduling. Our testbed model consisted of 576 activity types, 6 resources, and on average 6 state, resource, and precedence constraints per activity. In this application we allowed maintenance requests to request either refurbishment of specific subsystems or major systems. In order to schedule the maintenance requests the ASPEN system used a forward sweeping greedy dispatch algorithm which used strong knowledge of the precedences of activities in the plan. The resulting scheduler has been able to generate schedules for refurbishment problems involving approximately half of the subsystems (8 subsystems, 358 activities) in 8 minutes.

---

[6] Which targets the development of technologies enabling highly reusable, low-cost space transportation systems [12, 13].

[7] In comparison, the space shuttle refurbishment process takes approximately 65 days with a flight frequency of once per 4 months; the current Reusable Launch Vehicle initiative has a targeted flight frequency of once every 1-2 weeks.

[8] Developed by Rockwell International during the Phase 1 competition of the Reusable Launch Vehicle Program which ended in July 1996.

4 RELATED WORK

The idea of an application framework for planning/scheduling was pioneered in the OZONE system of Smith et al. [16, 17], which has been used in production management, transportation scheduling, and logistics applications. Differences between OZONE and ASPEN include the following:

- OZONE has emphasized applications in manufacturing and transportation planning and scheduling, while ASPEN is designed for spacecraft operations domains.

- OZONE emphasizes decision support tools, while ASPEN (due to the nature of spacecraft operations domains) emphasizes tools that support more autonomous decision-making applications.

5 CONCLUSIONS/FUTURE WORK

In this paper, we have described ASPEN, a reconfigurable, modular framework for planning/scheduling applications, and described three current applications of ASPEN in spacecraft operations. Although the development of a generic software architecture has required a substantial, initial investment of effort, we expect the total development effort for a set of scheduling applications to be significantly decreased (as compared to individually developing each of the applications).

ASPEN is currently a scheduling-oriented system, although some planning capabilities are supported for hybrid planning/scheduling applications such as the EO-1 and UFO-1. We plan to extend ASPEN to support additional planning capabilities. Currently, ASPEN already supports much of the functionality of state of classical planning systems [3]. We plan to extend ASPEN's planning capabilities so that it can be used as a framework for planning applications that also exploit the additional temporal reasoning and resource management capabilities which are available through ASPEN's scheduling-oriented facilities.

ACKNOWLEDGMENTS

This work was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract to the National Aeronautics and Space Administration. We thank Steve F. Smith for helpful comments on a draft of this paper. Robert Sherwood implemented spacecraft operations models for the ASPEN applications, and Quoc Vu implemented the GUI.

References

[1] Ultra high frequency follow on communications satellite system fact sheet, *http://www.fafb.af.mil/uhf/fact_sheet.html*.

[2] Pluto express information page, *http://www.jpl.nasa.gov/pluto/*, 1996.

[3] J. Allen, J. Hendler, and A. Tate. *Readings in Planning*. Morgan Kaufmann, 1990.

[4] 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.

[5] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[6] M. Fox. Isis: a retrospective. In M. Zweben and M. Fox, editors, *Intelligent Scheduling*. Morgan Kaufmann, 1994.

[7] S. Graves. A review of production scheduling. *Operations Research*, 29(4):646–675, 1981.

[8] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, 1969.

[9] S. Minton, M. Johnston., A. Philips, and P. Laird. Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1988.

[10] N. Muscettola. Hsts: Integrating planning and scheduling. In M. Zweben and M. Fox, editors, *Intelligent Scheduling*. Morgan Kaufmann, 1994.

[11] N. Muscettola, B. Smith, S. Chien, C. Fry, K. Rajan, S. Mohan, G. Rabideau, and D. Yan. On-board planning for new millennium deep space one autonomy. In *Proceedings of IEEE Aerospace Conference*, pages 303–318, Snowmass, CO, 1997.

[12] NASA. *Proceedings of the Technical Interchange Meeting on Highly Reusable Space Transportation*, Huntsville, AL, July 1995.

[13] NASA. *Proceedings of the Technical Interchange Meeting on Highly Reusable Space Transportation*, Huntsville, AL, August 1996.

[14] W. Pree. *Design patterns for object-oriented software development*. ACM Press, 1995.

[15] G. Rabideau, S. Chien, T. Mann, J. Willis, S. Siewert, and P. Stone. Interactive, repair-based planning and scheduling for shuttle payload operations. In *Proceedings of IEEE Aerospace Conference*, pages 325–341, Snowmass, CO, 1997.

[16] S.F. Smith and O. Lassila. Configurable systems for reactive production management. In E. Szelke and R. Kerr, editors, *Knowledge-Based Reactive Scheduling, International Federation of Information Processing (IFIP) Transactions B-15*. 1994.

[17] S.F. Smith, O. Lassila, and M. Becker. Configurable mixed-initiative systems for planning and scheduling. In A. Tate, editor, *Advanced Planning Technology*. AAAI Press, 1996.

[18] D. Speer, P. Hestness, M. Perry, and B. Stabnow. The new millennium program eo-1 mission and spacecraft design concept. In *Proceedings of IEEE Aerospace Conference*, volume 4, pages 207–227, Snowmass, CO, 1997.

[19] M. Zweben, B. Daun, E. Davis, and M. Deale. Scheduling and rescheduling with iterative repair. In M. Zweben and M. Fox, editors, *Intelligent Scheduling*. Morgan Kaufmann, 1994.

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