Automated Planning and Scheduling for Orbital Express (151)

Russell Knight
Jet Propulsion Laboratory, California Institute of Technology
russell.knight@jpl.nasa.gov

Abstract

The challenging timeline for DARPA's Orbital Express mission demanded a flexible, responsive, and (above all) safe approach to mission planning. Because the mission was a technology demonstration, pertinent planning information was learned during actual mission execution. This information led to amendments to procedures, which led to changes in the mission plan. In general, we used the ASPEN planner scheduler to generate and validate the mission plans. We enhanced ASPEN to enable it to reason about uncertainty. We also developed a model generator that would read the text of a procedure and translate it into an ASPEN model. These technologies had a significant impact on the success of the Orbital Express mission.

1. Introduction

Most technology missions have the challenge of discovering the limits and capabilities of new systems. This introduces planning challenges in that pertinent information needed for planning is not available until the various technology experiments are performed. But, it is often the case that bounds of performance are known a priori. Automated planning systems for these missions need to address this uncertainty at two points in the planning cycle: the long term plan and the short term plan.

The long term plan is produced before any of the experimental/unknown information is learned. The role of the long term plan is to ensure that enough resources are reserved ahead of time. Planning at this point requires accommodation of the bounds of possible execution.

The short term plan is produced immediately before execution, and some of the experimental/unknown information is known and should be integrated. This allows us to free resources and helps us to reduce cost or reduce the risk of other missions using the shared resources.

The two technologies we present here are 1) schema-level uncertainty reasoning (for long-term planning) and 2) procedure parsing for model generation (for short-term planning).

DARPA's Orbital Express mission demonstrated on-orbit servicing of spacecraft. Two spacecraft were flown: Boeing's ASTRO spacecraft, whose role was that of a doctor, and Ball Aerospace's NextSAT spacecraft, whose role was that of a patient. Experiments included rendezvous and capture, fluid propellant transfer, and on-orbit repair. Most of the planning for the mission was performed by the Boeing team, who also serviced requests from Ball Aerospace. The team was broken up into two units, the Rendezvous Planners who concerned themselves primarily with computing the locations and visibilities of the spacecraft, and the Scenario Resource Planners (SRPs), who were concerned with assignment of communications windows, monitoring of resources, and sending commands to the ASTRO spacecraft. The SRP position was staffed by JPL personnel who used the ASPEN planner scheduler. We report here on the technologies used for the SRP position.

2. Objectives

We had two primary objectives for Orbital Express: 1) evaluate scenarios for feasibility early in the design of the mission, and 2) provide responsive communications and commanding planning and scheduling during the mission. To satisfy both objectives, we modeled the mission scenarios using the ASPEN [5] planning system. OE required evaluation of many alternatives, so ASPEN was modified to accommodate reasoning about schema-level uncertainty. Rehearsals for operations indicated that the SRP needed to be very responsive to changes in the procedures. To accommodate this, we implemented a system for reading the procedures and interpreting these into ASPEN models.

3. Research Goals

The research goals we addressed were 1) schema-level uncertainty reasoning and 2) procedure parsing for model generation. Schema-level uncertainty reasoning has at its core the basic assumption that certain variables are uncertain but not independent. Once any are known, then the others become known. This is important where a variable is uncertain for an action and many actions of the same type exist in the plan. For example, the number of retries to purge the pump lines were unknown (but bounded), and each attempt required a sub-plan. Once we knew the correct number of attempts required for a purge, it would likely be the same for all subsequent purges.

To accommodate changing scenario procedures, we ingested the procedures into a tabular format in temporal order, and used a simple natural language parser to read each step and derive the impact of that step on memory, power, and communications. We then produced an ASPEN model based on this analysis. That model was tested and further changed by hand, if necessary, to reflect the actual procedure. This resulted in a great savings in time used for modeling procedures.

3. Schema-level Uncertainty Reasoning

To accommodate schema-level uncertainty reasoning in ASPEN, we modified the ASPEN Modeling Language (AML) to include a new reserved word – "uncertain". Any parameter of any activity type that was unknown (but bounded) would be labeled using this reserved word, e.g., uncertain int retries = [1,5] or uncertain string mode = ("idle", "transmitting", "off").

Then, when an instance of ASPEN was started with uncertain variables, the cross-product of the instantiations of uncertain variables was used to produce unique instances of plans. Each of these instances is called an alternative. Note that this is the cross product of the schema-level instantiations, not the actual activity-level instantiations. If we take our previous example, we would instantiate six alternatives:

retries = 1, mode = "idle"
retries = 1, mode = "transmitting"
retries = 1 , mode = "off"
retries = 2, mode = "idle"
retries = 2, mode = "transmitting"
retries = 2 , mode = "off"

Now, every activity in each alternative would have the same value, so it wouldn't matter how many activities we had. This differs greatly from activity-level uncertainty. In this case, we would need to generate an alternative for each possible activity assignment. This means that we would have exponentially many alternatives with increasing activities. Since the uncertain parameters are those that we expect to learn (and to not vary), then we can expect that if a parameter has a value earlier in the day, it will have the same value later in the day.

Also, operations staff was loathe to trust a more analytical and compressed form of uncertainty reasoning. It was a very compelling case to see all possible executions, and when they needed to justify why a certain resource allocation they found it simple and intuitive to use the set of alternatives.

To perform planning, we repair each alternative as if it were a separate schedule, and then perform a merge of the schedules. This results in what operations people consider to be "odd" schedules, where we might ask for resource allocations that are impossible for a single spacecraft but still must be accommodated if we are to accommodate all possible alternatives. If we are not granted an allocation, we can go to each alternative and either try to replan it or simply carry it as a risk.

3. Procedure Parsing for Model Generation

To accommodate late changes in procedures we implemented software that read procedures and produced ASPEN models. At first, this seemed like a daunting problem: we are in essence reading English text for content and producing a declarative activity/timeline based model of the procedure. One key observation we made is that the language of procedures is nearly as strict as a programming language, so we did not need to produce a parser capable of Natural Language Processing; we just needed to accommodate stylistic differences between authors. Of course, some free-form text does appear in the procedures, and the model needed to be annotated such that the ASPEN model parser would complain in a meaningful way and the human modeler would address the text that was not understood.

The procedures consisted of an introduction of human readable text, followed by a table of steps. They were authored using Microsoft Word. We found that most of the information needed to generate the procedure model was available in the table, so we would copy and paste the table into a Microsoft Excel

document. Our parser was written in Visual Basic, and embedded in the Microsoft Excel document.

Each step of the procedure had a number, the position or role responsible for carrying out the step, the action that was taking place, the response or verification to the action, and the expected duration. By parsing the action, we could determine whether the step included loops, if statements, or commands.

Loops in the procedures were accommodated using recursive decompositions. In ASPEN, it is often convenient to model activities and sub activities in trees known as hierarchical task networks. This representation is handy, but does not accommodate dynamic structures in the hierarchy. But, it does allow for disjunctions, e.g., an activity heater_parent can decompose into either a heater_child or a dummy activity. If we allow loops in the hierarchy, we can represent dynamic structures. The problem introduced by this is that the hierarchy appears to be infinitely deep. Therefore, we need to ensure that there are termination criteria, that is, at some point the loop breaks out to a sub-branch that has no loops.

If statements were modeled using disjunctive decompositions.

Both loops and ifs were candidates for uncertain variables.

The table also had commands that were to be sent to the spacecraft at execution time. Some of these commands were simple in that no further information was needed. In this case, the command activity was included as part of a decomposition. But, some of the commands required information to be input or computed. In this case, a human modeler needed to decide on the information source. To keep this from accidentally generating a working model, we would assign a known non-existent variable the string value of the text describing the command argument.

To ensure that command arguments and mnemonics were correct, we produced an ASPEN model from the command dictionary stored in a Microsoft SQL Database. This was a piece of SQL code written by Boeing personnel. This included the legal bounds for each argument.

If any procedure had poorly formed commands, the ASPEN parser would catch them, and the procedure would be corrected. This was a relatively free value-added effect that resulted in the correction of many procedures.

4. Impact

We were able to produce several alternatives for long-term planning so that enough communications resources were available at the time of execution. We also were able to deliver operations plans daily, even in the face of changing procedures and changing resource availability. Together this contributed to the success of the mission.

The overall affect of using ASPEN has been approximated by the flight director as a 26% reduction in the execution time of the mission, a 50% reduction in the daily staff required to produce plans, and a 35% reduction in planning errors. In fact, no miss-configured command was sent during operations.

9. Acknowledgments

The research described in this presentation was carried out by the Jet Propulsion Laboratory, California Institute of Technology, and was sponsored by the Defense Advanced Research Projects Agency.

Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government, or the Defense Advanced Research Projects Agency, or the Jet Propulsion Laboratory, California Institute of Technology.

10. References

[1] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davies, D. Mandel, S. Frye, B. Trout, S. Shulman, D. Boyer, “Using Autonomy Flight Software to Improve Science Return on Earth Observing One,” Journal of Aerospace Computing, Information, and Communication . April 2005.

[2] S. Chien et al., Automated Scheduling and Planning Environment (ASPEN), project home page, ai.jpl.nasa.gov, 2003.

[3] T. Estlin, T. Mann, A. Gray, G. Rabideau, R. Castano, S. Chien, and E. Mjolsness, An Integrated System for Multi-Rover Scientific Exploration, National Conference on Artificial Intelligence, Orlando, Florida, August 1999.

[4] Goddard Space Flight Center, EO-1 Mission page: http://EO-1.gsfc.nasa.gov

[5] G. Rabideau, R. Knight, S. Chien, A. Fukunaga, A. Govindjee, "Iterative Repair Planning for Spacecraft Operations in the ASPEN System," International Symposium on Artificial Intelligence Robotics and

Automation in Space, Noordwijk, The Netherlands, June 1999.

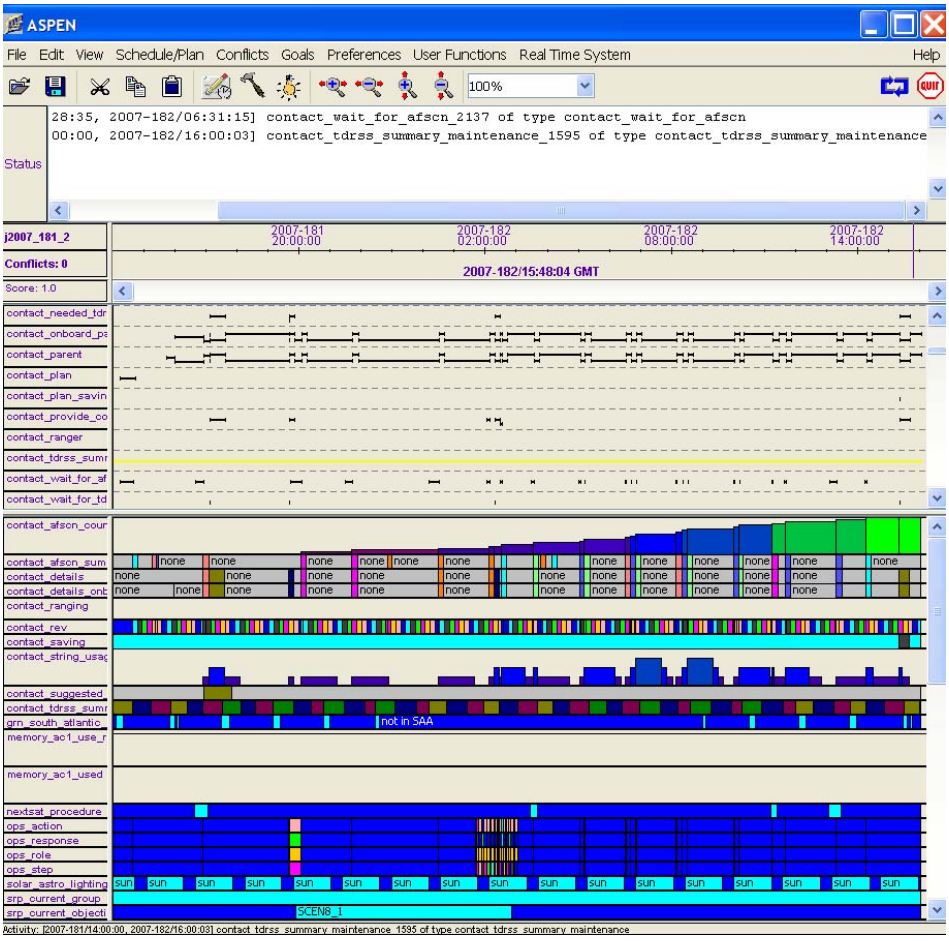

Figure 1. The ASPEN GUI