Embedding planning technology into satellite systems

David Kortenkamp
Pete Bonasso
TRACLabs Inc.
Houston TX
{korten,bonasso}@traclabs.com

David J. Musliner
Michael J. S. Pelican
Jesse Hostetler
SIFT LLC
Minneapolis MN
{dmusliner,mpelican,jhostetler}@sift.net

I. Introduction

There is an increasing need to develop on-board autonomy for satellite systems, both to increase their productivity and to protect them from hazards and threats such as component faults, approaching space debris, and dangerous space weather. We are developing an integrated system that demonstrates solutions to many of the challenges inherent in developing embedded planning systems for satellites. The Highly Autonomous Mission Manager for Event Response (HAMMER) system is designed to allow a satellite to operate and respond to threats even when it is not in communication with the ground or when time constraints require immediate response to threats. The HAMMER system attempts to meet mission objectives even in the face of threats. HAMMER prioritizes multiple, competing user goals and requests and determines an optimal ordering of satellite tasks to conserve resources and maximize capability. End user goals and requests are expected to come to the satellite asynchronously as the satellite is operating. Thus, new task schedules will need to be generated on-the-fly. Threats are also expected occur asynchronously and require on-the-fly replanning to counter the threats and still attempt to meet mission objectives. User requests will be at a high level (e.g., take an image of location X by time Y and download to location Z) and will need to be turned into a detailed plan of low-level satellite actions. The tight coupling between end user goals, mission planning, threat response, and task execution is a key challenge for these systems.

HAMMER integrates an on-board execution system with two different on-board planning and scheduling systems: a Mission Planner (MP) that plans optimal sequences of actions to achieve mission goals, and a Threat Response Planner (TRP) that refines the mission plan with pre-planned responses to threats. The benefits of the HAMMER system are that it: 1) can receive high-level end user goals and produce optimal satellite plans; 2) can respond to threats without ground intervention; 3) is scalable and reusable because the core components are model-driven.

I.A. Capabilities

Embedded planning for satellites needs several capabilities, including:

Reusability — The system should be implemented as a set of general purpose reasoning "engines" that execute on clearly-defined models of the system. This results in plan libraries that are largely reusable, because domain knowledge is captured in a satellite-independent fashion with specialization done at the spacecraft model and executive level.

Guaranteed Safety — By reasoning explicitly about potential threatening events and planning real-time safety-preserving reactions, the system should be able to guarantee that system safety is maintained whenever possible, while also exerting maximum effort to achieve mission goals. When goals must be sacrificed to preserve safety, the system should make those choices deliberately and automatically, and be able to explain its decisions to human designers and supervisors.

Scalability — A modular approach should allow multiple planners to to exist on different satellites. Thus, layers of planners could be added to focus on different mission aspects at different levels of abstraction. Pieces of the plans could then be sent to individual satellites for execution.

I.B. Example scenario

We consider a satellite in low Earth orbit, equipped with an imager with a shutter. The satellite's mission goals arrive dynamically from users on the ground, who request images of ground locations, taken during designated time windows and returned to designated ground stations before a deadline. These imaging goals may have different priorities, and the overall objective is to satisfy as many goals as possible. Satisfying an imaging goal requires the satellite to perform a series of actions such as powering up the imager, slewing the satellite to an appropriate viewing angle, determining when the satellite will be over the target location, opening the shutter, taking the image, storing the image, and downloading the image. Multiple end user requests can be pending at once, so the satellite actions for different imaging goals may need to be interleaved.

The satellite may encounter several different threats to both its mission and its safety, including:

- Solar radiation activity that harms the imager over time but still allows for imaging.
- Solar radiation activity that disrupts imaging and can quickly harm the satellite.
- Approach by a space debris object that threatens the physical integrity of the satellite.

Any threat can occur at any time. Detecting the specific event is not the focus of HAMMER, but is addressed by other work such as an enhanced neural-network based anomaly detector.[1] Threats can be countered by taking specific actions such as closing the shutter, slewing the satellite, or performing an orbital maneuver.

II. Embedded planning

The HAMMER architecture incorporates two planning systems with separate responsibilities: the Mission Planner generates a nominal plan from the user goals; the Threat Reaction Planner takes the nominal plan and builds rules that accomplish the plan while also responding to threats. We describe each of these in turn.

II.A. Mission planner

The Mission Planner (MP) has a model of the possible satellites actions, the conditions under which these actions are applicable, the consequences of the actions on the satellite state, and the resources they require. For example, in the scenario described above the MP has a model of the "take image" action, indicating that the action requires the imager to be on and the shutter be open before the action can happen, and that the action consumes memory resources. The MP also has a model of the possible states of the satellite and its environment. In our example scenario, this model includes the status of the shutter (open or closed), the imager (powered on or off), the satellite's orbit, and the amount of fuel.

II.A.1. Modeling language

There are many different planning modeling languages. Some are specific to single planners; for example, the NDDL modeling language is only used by EUROPA.[2] Other languages are generic and are used as input to many different planning systems. For example, the Planning Domain Description Language (PDDL) is used for planning competitions.[3] In this work, we used the Action Notation Modeling Language (ANML) that is being developed jointly at NASA JPL and NASA Ames Research Center.[4] We chose ANML because of its NASA roots in modeling space systems, because of its ease in expressing temporal constraints, and because it can be imported by several NASA planning systems that have flight heritage.

ANML uses a C-style syntax. The basic elements of the language are statements and declarations; one can declare actions, fluents, constants, and (structured) types. The built-in types are booleans, integers, floats, strings, objects, and symbols. The last two are *open types*, all other built-in types are *closed types*. Statements come in two forms: *effects* and *constraints*. *Effects* define the *transition model*, and *constraints* put bounds on inference about the transition model. ANML semantics are similar to standard state-transition models; however, there are complications due to instantaneous transitions and continuous time. The fundamental entities are *fluents*: variables that change over time. A *state*, S, is just an instantaneous assignment to all fluents; and state *trajectories*, $\mathcal{S}$, are an assignment of values to all fluents for intervals of time. All statements exist within a *temporal scope*. For example, a temporally unqualified constraint within an action must hold over the entire execution of the action; similarly, an unqualified effect happens over the entire action. One can specify points, open intervals, closed intervals, and half-open intervals (both kinds). Existential temporal intervals can be constrained with respect to each other.

II.A.2. Modeling the domain

We modeled our simple satellite scenario in ANML in order to verify that we could express the key actions and constraints of a satellite domain. For example, the ANML to describe actions that open and close the satellite shutter is:

```
action open_imager_shutter() {
  duration := 1;
  [all] shutter_position == CLOSED :-> OPEN;
}

action close_imager_shutter() {
  duration := 1;
  [all] shutter_position == OPEN :-> CLOSED;
}
```

Similarly, to power on and off the imager:

```
action power_on_imager() {
duration := 60;
[all] imager_state == OFF :-> ON;
}

action power_off_imager() {
duration := 10;
[all] imager_state == ON :-> OFF;
```

A slightly more complicated example is slewing the satellite to point at a particular location, which shows the use of resources and calls out to external procedures for doing domain-specific computations:

```
action point_at_location(Location location) {
  [start] {
    constant Attitude
      cAtt := attitude,
      gAtt := location_to_attitude(location);
  }
  duration := swing_time(cAtt,gAtt);

  (all] attitude :-> gAtt;
  [all] fuel :consumes fuel_consumption(cAtt,gAtt);
}
```

Taking an image requires checking the amount of memory available before the action starts and then reducing the available memory by that amount after the action:

```
action take_image() {
  duration := 1;
  [start] {
    shutter_position == OPEN;
    imager_state == ON;
  }

  [all] attitude == [start] attitude;

  [start] memory >= size_of_image ;
  [end] memory :consumes size_of_image ;
}
```

II.A.3. Planning

In this work, we used an in-house planner called AP.[5] AP cannot read ANML directly, but does read in PDDL2.1,[3] which is very close to ANML syntax. We hand-translated each ANML construct into PDDL2.1 (we have since built an ANML to PDDL2.1 translator that automatically performs this translation). For example, the open shutter construct shown in the previous section becomes the following in PDDL2.1:

```
(define (durative-action open_imager_shutter)
    :duration 1.0
    :condition (at start (not (open shutter)))
    :effect (at end (open shutter))
    :comment "opens shutter.")
```

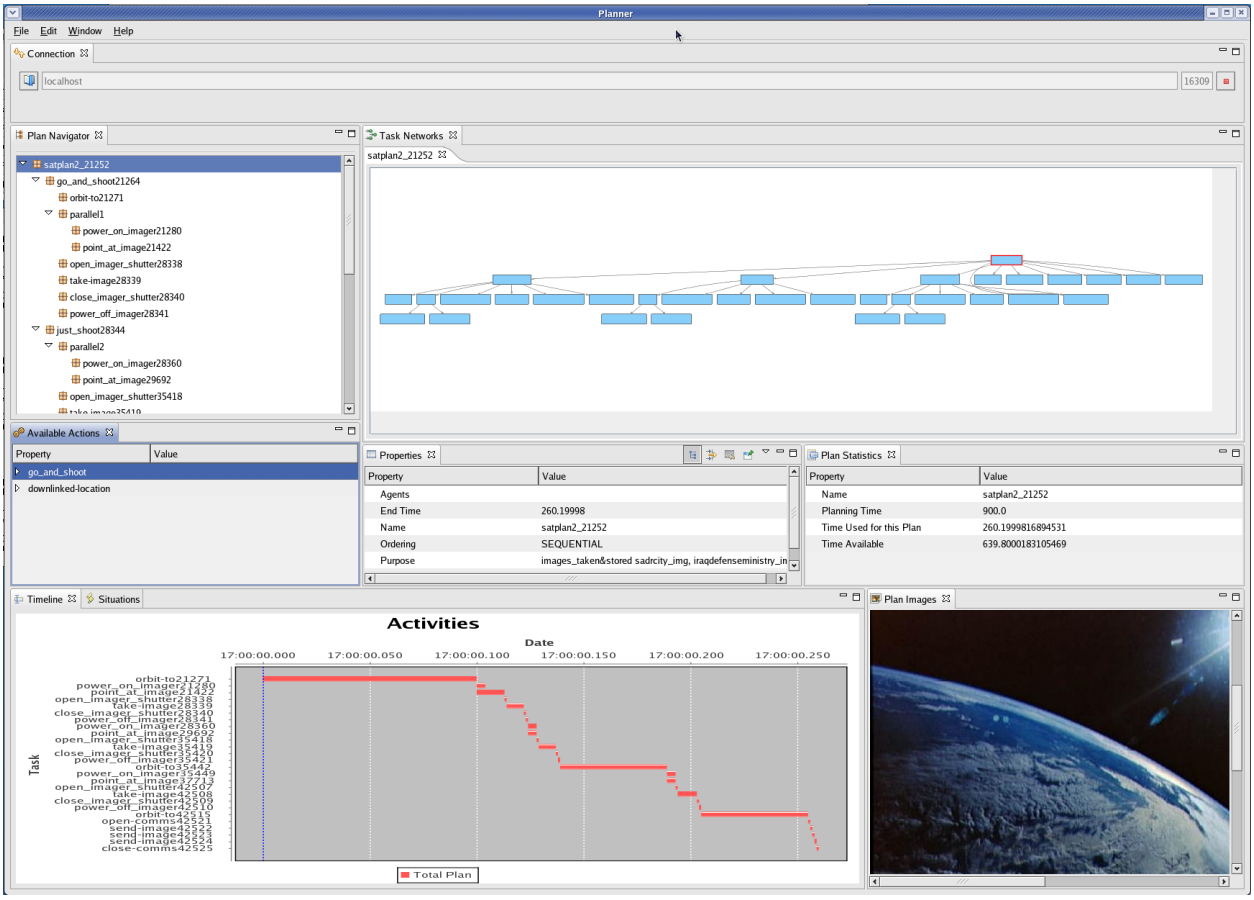

Figure 1. User interface for the mission planner.

Given a set of actions in ANML that have been translated to PDDL2.1, the planner can then accept a set of goals that need to be accomplished. The planner uses the actions to satisfy the goals. Figure 1 shows a graphical user interface of a plan produced. It is difficult to read at the resolution of a printed paper, but the upper right shows a graphical break down of the tasks that need to be performed and their relationship to higher-level tasks. The bottom left shows a timeline of all plan activities, including taking images and downlinking images. Figure 2 shows a closer view of the plan timeline. Note that the planner automatically turns the imager on and off at the right times and opens and closes the shutter.

II.B. Threat response planner

The Threat Response Planner (TRP) is responsible for taking the time-constrained sequence of actions generated by the MP, determining the potential impact of threats on that sequence, and generating a set of responses to those threats

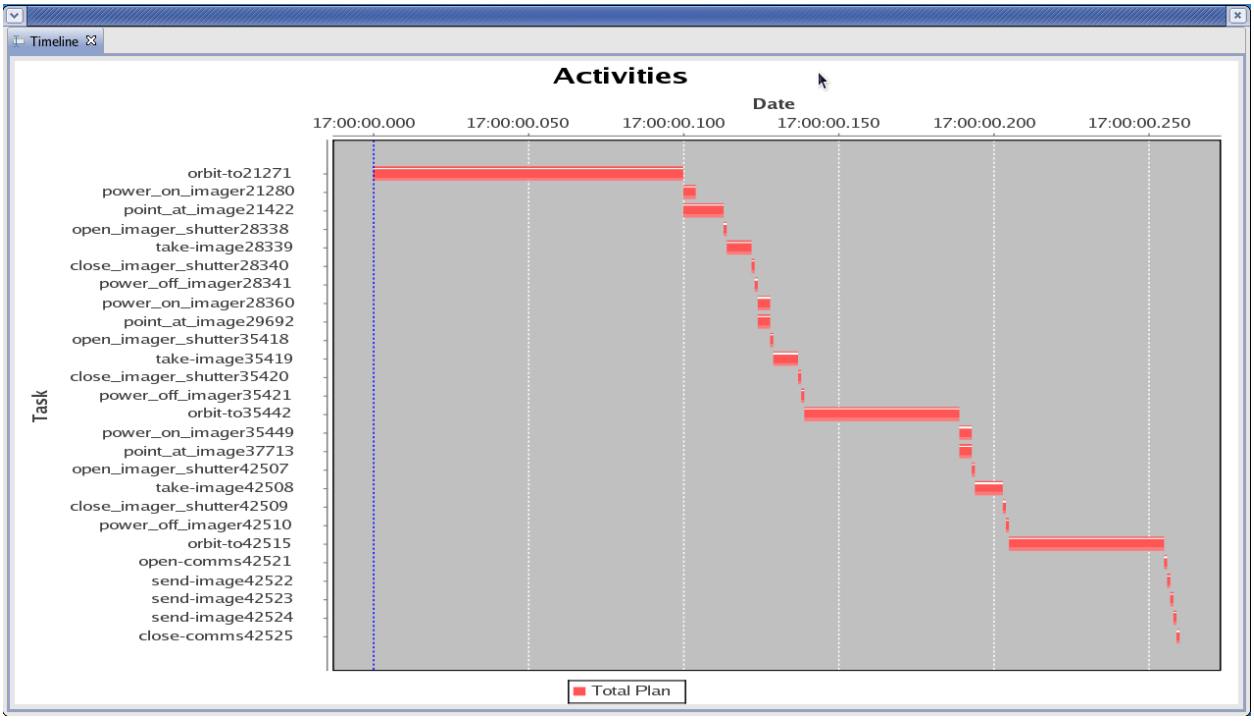

Figure 2. The plan produced by the mission planner.

that keep the satellite safe while still accomplishing the end user goals if possible. By expanding a state-transition model of possible futures, the TRP builds reactive plans that determine the best satellite actions that should be taken for every possible (reachable) future world state. Using its models of threats (e.g., what types of solar radiation are anticipated) and their effects on the satellite and its mission (e.g., how solar radiation disrupt the spacecraft), the TRP reasons about the possible interleavings of goal-driven actions (e.g., preparing to capture an image) and safety-driven reactions (e.g., closing the camera shutter to avoid damage). The TRP automatically plans safety reactions to prevent any catastrophic failures, and includes goal-achieving reactions whenever they do not endanger system safety. The TRP uses automatic model checking techniques to formally prove that the resulting reactive plan will maintain system safety while making best-effort attempts to achieve the mission plan laid out by the MP.

The TRP either successfully generates a safety-preserving plan, which can then be downloaded to the executive to be run, or it finds that safety cannot be guaranteed. In that case, the MP may choose to alter the Mission Plan to vary the goals it is trying to achieve, the threats it is having the TRP consider, or other aspects of the problem. As with the MP, the TRP is a domain-independent planning system that can be easily used on-board different satellites, having different actuators, sensors, threats and goals. In this work, we used the Cooperative Real-Time Control Architecture (CIRCA)[6,7] to provide this capability. CIRCA is a planning and control system that can devise safety-preserving plans in the face of unpredictable events and execute those plans with hard real-time guarantees.

II.B.1. Modeling for Reaction Planning

The CIRCA system includes a Threat Response Planner (TRP)[a] which devises a plan to accomplish mission goals while avoiding failures and generates the specification of a controller that can execute its plan. The first step in using the CIRCA system is to describe the reaction planning problem in CIRCA's domain modeling language. The problem description specifies the initial state of the world, a set of goal states which the planner attempts to reach, a set of failure states which the planner must avoid, and a set of transitions that move the world from one state to another. State transitions can be of one of four types:

Event — A transition triggered by an occurrence in the environment that is outside of the control of the planner. Events allow us to model unpredictable threats to the safety of the satellite such as a weather event or a compo-

[a] Also called the Controller Synthesis Module in some publications.

nent failure.

Temporal — A transition that may occur any time after a minimum delay, provided its preconditions remain satisfied. Temporal transitions represent the passage of time and the execution of uncertain processes: the transition may occur, or it may not. Temporal processes may be interrupted by negating any of their preconditions. Temporal transitions are often used to represent transitions to the distinguished failure state, meaning that after some time, a process may lead to catastrophic failure. The TRP's job is to build reactive plans that make such failures impossible.

Reliable Temporal — Unlike normal temporal transitions, a reliable temporal transition has an upper time bound and is guaranteed to occur by that bound, if its preconditions remain satisfied.

Action — An action that the controller agent may execute. The TRP chooses actions to move from one state to another, either to accomplish goals or to avoid (prevent) failure states.

```
;; High-intensity solar radiation event may begin any time it isn't already on
(def-event high-intensity-solar-radiation-threat-begin
    :preconds ((high-intensity-solar-radiation-threat F))
    :postconds ((high-intensity-solar-radiation-threat T)))

;; If no action is taken, the solar radiation threat leads to failure
;; after 2 time units. The ``imager-close-shutter'' action
;; defeats this transition by making the ``imager-shutter-open''
;; precondition false.
(def-temporal high-intensity-solar-radiation-threat-damages-imager
    :preconds ((imager-health ok)
               (high-intensity-solar-radiation-threat T)
               (imager-shutter-open T))
    :postconds ((imager-health fail) (failure T))
    :min-delay 2)

;; Close the shutter on the imager
;; Worst-case-execution-time = 1 time unit
(def-action imager-close-shutter
    :preconds ((imager-power on) (imager-shutter-open T))
    :postconds ((imager-shutter-open F))
    :wcet 1)
```

Figure 3. A partial TRP domain specification, including a solar radiation event (represented by an *event* and a *temporal* transition to failure) and an *action* that can defeat the failure transition.

The TRP attempts to create a *safe* plan, which is a plan in which all failure states are unreachable. To do so, the TRP planner searches the state space to determine which transitions lead to a failure state and looks for actions that it can execute to "preempt" those transitions by violating their preconditions.

Figure 3 illustrates a typical threat model taken from the larger HAMMER problem domain described below in Section I.B. In this example, we specify the beginning of a solar radiation event as an *event*, indicating that it may occur at any time that its preconditions are met. The solar radiation event causes the preconditions of a temporal transition to a failure state to become satisfied. The temporal transition represents the shortest amount of time that it might take for the solar radiation event to cause damage to the satellite's imager. The planner can avoid failure by scheduling the "close-shutter" action to occur before this minimum time period elapses. Closing the shutter causes a violation of one of the preconditions of the temporal transition, thus avoiding a transition to a failure state.

Note that it is not always possible to capture an image in our example scenario. Capturing an image is not an instantaneous event, and the timing of solar radiation event could be such that the camera would be damaged before image capture is complete. CIRCA was able to produce a plan in this domain (see Figure 4) that protected the satellite from damage and captured an image if given the opportunity. The planner correctly determined that closing the camera shutter in response to an event (see Figure 5) would protect the camera from damage but that the camera should be kept powered on and pointed at the target in case the threat ended and it was safe to open the shutter again.

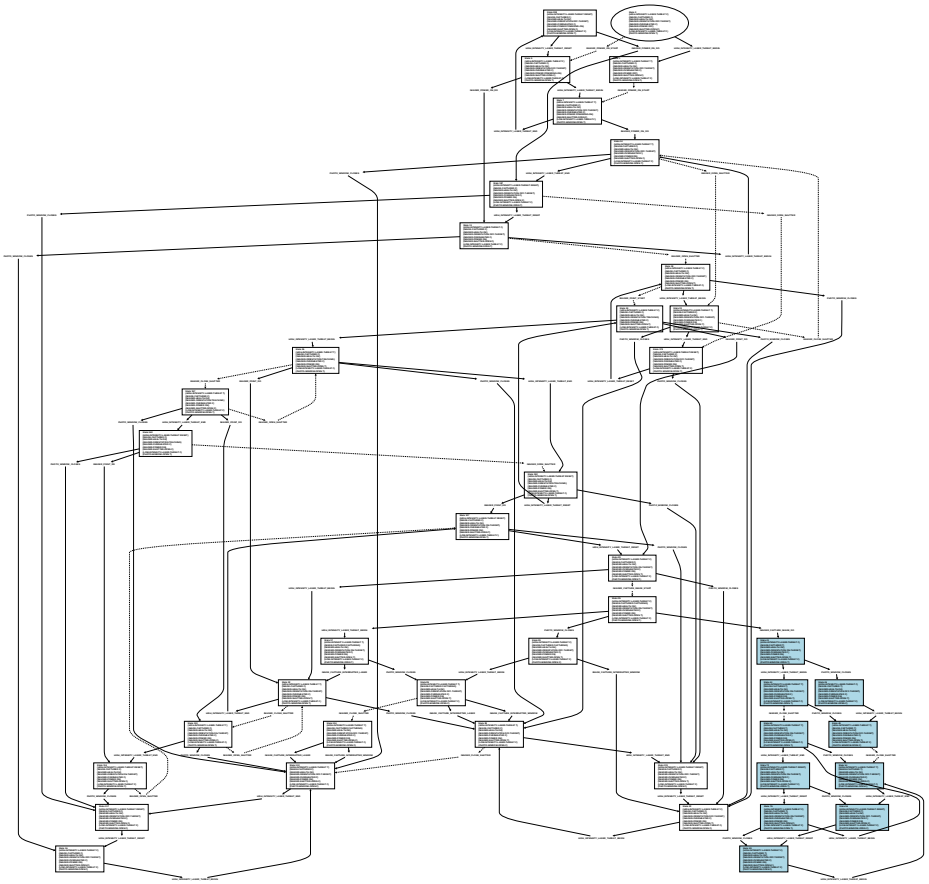

Figure 4. A state space graph illustrating CIRCA's plan to capture an image while responding to a solar radiation event. The blue squares are states in which the image has been captured successfully.

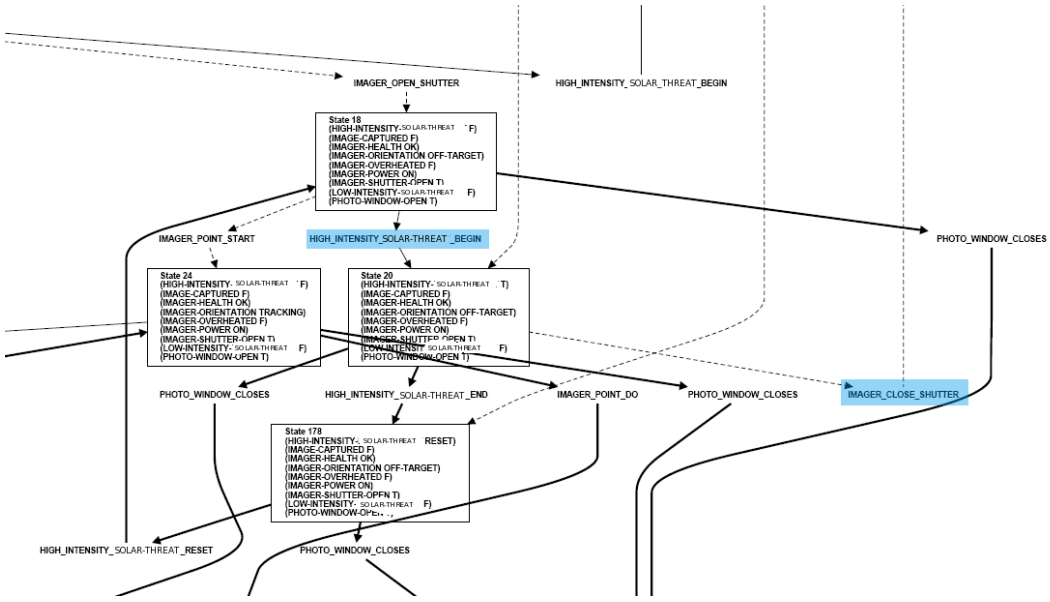

Figure 5. Detail of the CIRCA state space graph showing a threat reaction. Notice that in State 20, a solar radiation threat is present. The planner takes the `IMAGER_CLOSE_SHUTTER` **action (highlighted) to protect the satellite.**

CIRCA produced a safe plan for this domain with no other information than a list of the actions available to it and the events it might face. Nowhere in the domain model is there a rule of the form `IF solar radiation event THEN close-shutter`. The ability to derive multi-stage dependency relationships from mere declarative specifications makes CIRCA extremely adaptable to changes in the problem specification or even the application domain. If new capabilities are added to the satellite or new potential threats are discovered, CIRCA can accommodate them into its world model with only simple changes to the problem description. If, for example, we decide to use a new imager that requires a shorter time to capture an image, we can simply alter the execution time of the "capture-image" action in the problem description. The planner might then be able to get its image despite a solar radiation event by opening the shutter, snapping the image, and closing the shutter again before any damage is done. Without the intelligent planning capabilities of CIRCA, any change to the satellite's capabilities requires a parallel change and subsequent re-verification of the satellite's control system.

As systems become more complex and the relationships between components become more difficult to comprehend, it is increasingly likely that human control system designers will overlook subtle interactions that threaten the safety of the mission. CIRCA can guarantee that the system as modeled will be safe under all circumstances, either by generating a safe plan or by alerting designers that a safe plan is not feasible given the parameters of the system.

Similarly, CIRCA can expose inconsistencies in mission objectives. In this example, if we reformulate the problem to CIRCA so that not capturing an image constitutes a failure, CIRCA recognizes that it is impossible to guarantee mission safety. Because a solar-radiation threat could begin at any time and an image cannot be captured while a solar radiation event is present, there are possible sequences of events that lead to an unavoidable failure to capture an image. CIRCA is capable of recognizing such situations, and will report that no possible plan can satisfy all mission requirements. Human mission planners can then revise their mission plan to be internally consistent, perhaps by relaxing some mission requirements or by making changes to the flight plan that will alter the threat profile faced by the satellite.

II.B.2. Real-time Execution

Generating safe plans of action is only half the battle: these plans must be executed faithfully by the satellite's onboard control system. CIRCA's Real-Time Subsystem (RTS) is responsible for carrying out the plans generated by the TRP. The RTS is a memoryless, unclocked real-time control system. In execution, the RTS continuously loops over a sequence of test-action pairs (TAPs) fed to it by the TRP, each of which consists of a set of precondition tests and an action to execute if all preconditions are met. All TAP scheduling decisions are made and verified for safety offline by the TRP. The RTS's only job is to execute them. This simple runtime execution model allows the performance of the RTS executive to scale easily to highly complex problem domains while maintaining hard real-time guarantees.

III. Demonstration

We have developed a proof-of-concept demonstration of the HAMMER on-board planning system for satellites. This included a Mission Planner (MP) that took user goal requests to take images and generated a sequence of actions that accomplished the goals. The demonstration also included a Threat Response Planner (TRP) that generated a set of rules to implement the user goal requests while keeping the satellite safe from threats. The MP and the TRP did not directly exchange data. Instead, the generated plan from MP was hand-entered into the TRP. An executive was then used to run the plan against a simple, state-based simulation of a satellite.

To demonstrate the ability of the CIRCA RTS to control a satellite during mission execution, we created a simplified satellite simulation and a graphical interface to visualize its operation. The structure of the simulation mimics the usage of the RTS in a real system: the RTS receives information from its environment, either by actively querying sensors or by receiving notifications of state changes, and issues appropriate commands to carry out mission objectives and protect the satellite from threats. The interface between the RTS and the systems it controls is extremely generic: any operation that can be performed in C code can be controlled by the RTS.

Shown in Figure 6, the demonstration display component consists of a graphical interface implemented in Java that listens for updates to the state of the simulated satellite and displays them to the user. The interface also provides buttons that allow a user to initiate events that threaten the safety of the satellite and observe the responses enacted by the RTS. This capability provides a realistically unpredictable threat environment in which to test the RTS. As would be the case in an actual mission, the RTS has no way of knowing when or if a threat will present itself– the user can turn on the threat at any time.

Testing the RTS in this manner confirms that it is capable of protecting the satellite from unpredictable threats.

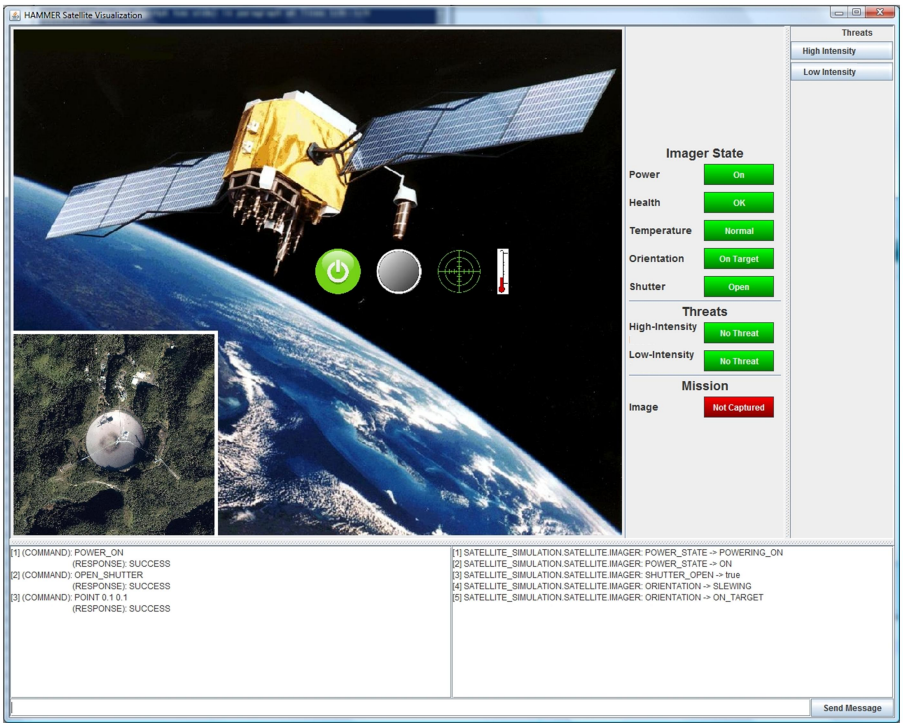

Figure 6. The HAMMER satellite demonstration display gives the user control over solar events and shows the status of various satellite components, sensors, and mission goals.

No matter at what stage in plan execution the events are initiated, the RTS responds appropriately in time to prevent damage. When the events pass, the RTS resumes its business of achieving mission objectives (acquiring images).

IV. Related work

Researchers have investigated several mission management architectures that combine planning, execution and event detection. Two of the most relevant architectures have been developed by NASA for unmanned spacecraft. JPL has developed a mission management system for both the TechSat2 and the EOS orbiting spacecraft.[8] This system conducts autonomous science on-board the spacecraft. NASA Ames Research Center has a long history of mission management architectures for unmanned spaceflight, starting with the Remote Agent.[9] This work has continued under the Automation for Operations (A4O) project, from which we draw the ANML and EUROPA technology being used in this proposal.[10] TRACLabs Inc.'s 3T architecture has also been used for several NASA mission management experiments, including human-rated test facilities.[11]

Planning is, of course, an active area of research in artificial intelligence. An excellent overview of planning in the space domain is given by Jonsson et al.[12] Different techniques include systems that transform planning problems into satisfiability problems such as SatPlan,[13] more popular graph search planners such as FF,[14] and knowledge-based approaches that underlie SIFT's Playbook hierarchical task network planner, SHOP2.[15] There are relatively few competing approaches to the reaction planning approach we are taking in the TRP. Wong-Toi's game-theoretic approach to generating supervisory controllers for systems modeled as linear hybrid automata[16] is similar in goal to our work on CIRCA, but uses algorithms that are inherently less scalable.

V. Future directions

Future work will focus on a tight integration between the on-board planning system and the on-board executive, as well as dealing with continuous planning and re-planning in the face of changing situations. We will also move to a more realistic simulation of satellite hardware and sensors. In addition, we will integrate the planning and execution system with orbital planners and event detection systems. Our goal is to develop a flight experiment on a satellite system, to validate the effectiveness of our approach to high-confidence on-board planning and execution.

VI. Acknowledgements

This work funded under Air Force Research Laboratory, Space Vehicles Directorate, contract FA9453-09-M-0093. The authors wish to thank Paul Zetocha of the AFRL, Phil Courtney of SRA International, and Derek Surka of Data Fusion and Neural Networks Inc. for their contributions to the ideas presented in this paper.

References

[1]Desrocher, D., Tschan, C. R., Koons, H. C., and Bowman, C. L., "Data Exploitation Techniques for Enhanced Satellite Operations," *Proc. SatMax 2002: Satellite Performance Workshop*, 2002.

[2]Frank, J. and Jónsson, A., "Constraint-Based Attribute and Interval Planning," *Constraints*, Vol. 8, No. 4, 2003, pp. 339–364.

[3]Fox, M. and Long, D., "PDDL2.1: An extension to PDDL for expressing temporal planning domains," *Journal of Artificial Intelligence Research*, Vol. 20, 2003, pp. 2003.

[4]Smith, D., Frank, J., and Cushing, W., "The ANML Language," *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation for Space (i-SAIRAS)*, 2008.

[5]Elsaesser, C. and MacMillan, R., "Representation and Algorithms for Multiagent Adversarial Planning," Tech. Rep. MTR-91W000207, The MITRE Corporation, 1991.

[6]Musliner, D. J., Durfee, E. H., and Shin, K. G., "CIRCA: A Cooperative Intelligent Real-Time Control Architecture," Vol. 23, No. 6, 1993, pp. 1561–1574.

[7]Musliner, D. J., Durfee, E. H., and Shin, K. G., "World Modeling for the Dynamic Construction of Real-Time Control Plans," *Artificial Intelligence*, Vol. 74, No. 1, March 1995, pp. 83–127.

[8]et al, S. C., "Autonomous Science on the EO-1 Mission," *Proceedings of International Symposium on Artificial Intelligence Robotics and Automation in Space (i-SAIRAS)*, 2003.

[9]Muscettola, N., Nayak, P. P., Pell, B., and Williams, B. C., "Remote Agent: to boldly go where no AI system has gone before," *Artificial Intelligence*, Vol. 103, No. 1, 1998.

[10]Frank, J., "Automation for Operations," *Proceedings of the AIAA SPACE 2008 Conference*, 2008.

[11]Bonasso, R. P., Kortenkamp, D., and Thronesbery, C., "Intelligent Control of a Water Recovery System," *AI Magazine*, Vol. 24, No. 1, 2003.

[12]Jonsson, A., Morris, P., Muscettola, N., and Rajan, K., "Planning in interplanetary space: theory and practice," *Proceedings of the Fifth International Conference on AI Planning and Scheduling (AIPS)*, 2000, pp. 177–186.

[13]Kautz, H., Selman, B., and Hoffmann, J., "SatPlan: Planning as Satisfiability," *Abstracts of the 5th International Planning Competition*, 2006.

[14]Hoffmann, J. and Nebel, B., "The FF Planning System: Fast Plan Generation Through Heuristic Search," *Journal of Artificial Intelligence Research*, 2001, pp. 253–302.

[15]Nau, D., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., and Yaman, F., "SHOP2: An HTN Planning System," *Journal of Artificial Intelligence Research*, 2003, pp. 379–404.

[16]Wong-Toi, H., "The Synthesis of Controllers for Linear Hybrid Automata," *Proceedings of IEEE Conference on Decision and Control*, Dec. 1997.