An Autonomous Control System for an Intra-Vehicular Spacecraft Mobile Monitor Prototype

Gregory A. Dorais, Salvatore D. Desiano[1] Yuri Gawdiak, and Keith Nicewarner[1]
NASA Ames Research Center
MS 269-2, Moffett Field, CA, 94035, USA
gdorais@arc.nasa.gov, sdesiano@arc.nasa.gov, ygawdiak@hq.nasa.gov, knicewar@arc.nasa.gov

Keywords Artificial Intelligence, Space Autonomy, Space Robotics, Planning, Diagnosis, International Space Station, Integrated Vehicle Health Management

Abstract

This paper presents an overview of an ongoing research and development effort at the NASA Ames Research Center to create an autonomous control system for an internal spacecraft autonomous mobile monitor. It primary functions are to provide crew support and perform intra-vehicular sensing activities by autonomously navigating onboard the International Space Station. We describe the mission roles and high-level functional requirements for an autonomous mobile monitor. The mobile monitor prototypes, of which two are operational and one is actively being designed, physical test facilities used to perform ground testing, including a 3D micro-gravity test facility, and simulators are briefly described. We provide an overview of the autonomy framework and describe each of its components, including those used for automated planning, goal-oriented task execution, diagnosis, and fault recovery. A sample mission test scenario is also described.

1. Introduction

The Personal Satellite Assistant (PSA) project is a NASA research and development activity to design an intelligent, small, free-flying, remote-sensing vehicle [1]. It is designed to autonomously navigating in 3-dimensions within a pressurized, micro-gravity environment, diagnosing systems in its environment, and interacting with people such that it is useful, easily understood, and easily commanded in a time-efficient manner. The primary operating environment target is the International Space Station (ISS), but other environments include the Space Shuttle and future manned spacecraft, such as one designed to carry a crew to the Moon or Mars. The PSA has various environmental sensors as well as audio/video human-interface devices. It can be remotely commanded at various levels of autonomy and can be commanded locally by simple speech commands and human gestures.

Mission Roles

The two primary mission roles for the PSA are to improve spacecraft crew productivity and to decrease mission risk by serving as part of an integrated spacecraft health management system.

Spacecraft health-management support role—the PSA will provide mobile monitoring, diagnosis, and communication capabilities. The PSA is being designed to supplement the spacecraft's Environmental Control Life and Support System (ECLSS) by measuring temperature, pressure, humidity, and various gas levels (e.g., oxygen, CO_2) and recording a visual log as it traverses the spacecraft. The PSA will help diagnose and calibrate spacecraft sensors, temporarily replace faulty environmental sensors, generate acoustic, temperature, and gas concentration maps, locate gas and fluid leaks, filter atmospheric particles, as well as characterize heat sources with its infrared camera.

Crew productivity role—the PSA will provide several support capabilities including: remote visual monitoring and task recording, video and data display, payload & core system knowledge management, inventory tracking, just-in-time training, and standard PDA functions (schedule, notes, activity lists, calculations, etc.). These capabilities will directly support flight crews in the daily execution of payload experiment and core system tasks.

To support the flight crews, ground crews, and payload scientists, the PSA can be used for monitoring and communication using its audio and video sensors as well as perform videoconferencing and display a variety data on its LCD screen. The PSA will allow ground crews and scientists to be virtually located inside the spacecraft. Moreover, the PSA's autonomy capabilities will allow remote users to interact with the crew and spacecraft in a human-centered way while providing real-time data collection and communication.

The ISS, for example, is an extremely ambitious operational environment for the crew (3-6 members) with tens of thousands of inventory items to track and hundreds of experiments to manage covering a wide spectrum of science disciplines. A PSA could be used to increase the productivity of the ISS by automating or otherwise reducing the crew time required to perform tasks as well as by enhancing or enabling science activities that would otherwise not be performed due to insufficient crew time.

This paper describes select PSA high-level functional requirements, prototypes and their test facilities, autonomy technology components, and a test mission scenario.

2. Functional Requirements

The following are a few of the higher-level functional requirements that may be of particular interest.

Requirement 1—create a self-contained portable device with environmental sensors, computational capabilities to analyze the sensor data and perform diagnoses, and a video display. The sensors are to function inside the ISS and similar operating environments. The high priority sensors include those that measure local temperature,

[1] QSS Group, Inc.

atmospheric pressure, humidity, and gas concentrations including O_2 and CO_2. Lower priority sensors include visible-light still and motion cameras, thermal imager, Geiger counter, NDIR spectrometer, electromagnetic detector, RFID tag detector for inventory management, microphone, and a directional acoustic detector array for localizing emissions.

Requirement 2—stamp the sensor data with the time and a 6-DOF position of the sensors relative to the environment. The 6 degrees-of-freedom (DOF) correspond to X, Y, Z translations and yaw, pitch, roll orientations relative to a global origin. Satisfying the position element is of this requirement is challenging retrofitting the ISS with active beacons to create a local GPS system is strongly discouraged except for special cases. Our current approach is to develop a system that can do self-localization using a combination of stereo-cameras to build depth maps and sense motion by means of optic flow algorithms and fuse this with data from a 6-DOF inertial measurement unit (gyros and accelerometers), and proximity sensors. As necessary, we can mitigate risk by engineering the environment with passive fiducial marks as needed.

Requirement 3—"station-keep" on command by maintaining a fixed position and orientation relative to its environment. This capability is particularly important since the system has no "brakes." Moreover, many tasks require maintaining a fixed position for a period of time. Note that the environment, i.e., the ISS, is continually in motion as it orbits the Earth and performs minor attitude adjustments.

Requirement 4—navigate to various positions on command, avoiding static and dynamic obstacles. This requirement can be viewed as a corollary of requirements 2 and 3. If the system already has the sensors, controllers, and actuators to determine its absolute position and maintain it, enabling it to navigate requires no additional hardware. Allowing the system to navigate to various positions again increases the flexibility of the system while decreasing the crew time required to perform a task. For example, searching for a leak or a measuring gas concentrations throughout a module can be quite time-consuming. The task is more efficient if it doesn't require a crewmember to be present even if the task takes longer.

Requirement 5—minimize the time required by the crew to operate the system while enabling crewmembers to command the system at the level of autonomy they desire. This requirement is in keeping with the general principle that crew time is extremely valuable. In some cases, this means that totally autonomous systems are preferable to manual systems. However, there are cases where autonomous systems require more crew time because the overhead in figuring out how to command the system to perform a task autonomously is greater than doing the task manually. Consequentially, the requirement is essentially for the system to be adjustably autonomous. If necessary, another system, such as the environmental life support system can command it to localize a heat source without requiring any crew intervention. In another task, a crewmember can command it to go to a certain location and notify him upon arrival, at which time the crewmember teleoperates the system as desired. In order to achieve this requirement, the system must have mixed-initiative planning, scheduling, and execution capabilities, and the ability to effectively communicate with the human operator so the operator understands what the system is doing and why it is doing it, and the system can interpret what the operator wants and can translate it into commands it can execute.

Requirement 6—perform continuous active hybrid temporal-variable diagnostics on its environment and equipment in it. We define a diagnostic system here to be one that determines the sets of likely system states that are consistent with the observations and the model of the system. A temporal-variable diagnostic system can use observations that change over time, e.g., recognize trends. A hybrid diagnostic system is one that can reason given both continuous-valued and discrete-valued observations. Typically, different approaches are used for continuous and discrete-valued observations, but many systems require that both be reasoned about simultaneously. An active diagnostic system is one that determines what additional observations are needed to disambiguate the state of the system being diagnosed. For example, consider a system with a HIGH-TEMPERATURE warning light that is on. Two possible diagnoses are that the system is indeed overheating or the temperature sensor is faulty. By verifying the temperature of the system with an independent measurement, such as can be provided by a mobile sensor, we can then determine more accurately which of these two diagnoses is more likely correct. One of the uses of this portable sensor device is as part of a larger Integrated Vehicle Health Management (IVHM) system so having this diagnostic capability can increase the likelihood of early detection and accurate diagnosis of problems without requiring crew time.

Although there are several other requirements, these six functional requirements effectively constrain the space of possible solutions. Other notable requirements involve safety, reliability, and ease-of-use. In particular, a smaller overall size and longer operation between recharges is better.

3. PSA Prototypes and Test Facilities

The PSA project is using an iterative, rapid prototyping approach for the development of the hardware. We began by developing a 4-fan prototype with a stereo camera pair that floats on a thin cushion of air over a granite table. This PSA Model 1 prototype was capable of navigating in three degrees of freedom (DOF): X, Y, and yaw. The stereo camera were used to estimate position and velocities.

PSA Model 2

While the Model 1 was being tested, a 6-DOF Model 2 prototype was developed and is shown in Figure 1.

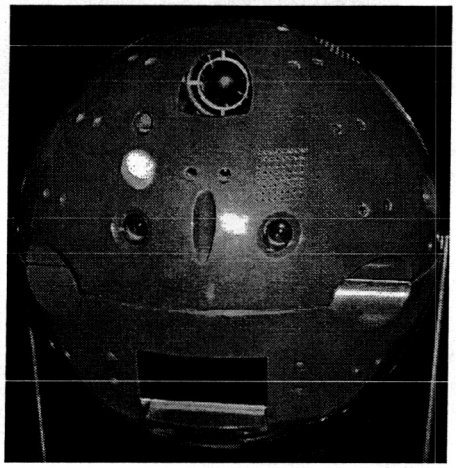

Figure 1 - PSA Model 2

The Model 2 is 12" in diameter (the targeted flight model diameter of 8") and capable of position and velocity estimation and motion in 6-DOF (X, Y, Z, yaw, pitch, roll) 6-DOF position and velocity estimation is achieved using multiple stereo-pair cameras (between 1-4 pairs). Propulsion and attitude control 6-DOF are achieved using 6 fan pairs located in 6 ducts. The Model 2 has a 3.8" diag. LCD located at the center of its front lower hemisphere. The LCD can be used to display data generated locally as well as data received via its wireless network, e.g., text terminals, images, schematics, videos, and support teleconferencing. The location of these and additional components are depicted in Figure 2.

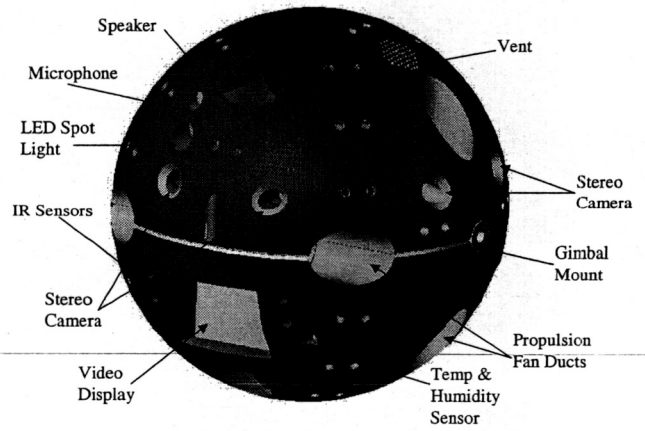

Figure 2 - PSA Model 2 Annotated Drawing

Micro-gravity Test Facility

To test the Model 2 on Earth, a micro-gravity test facility was developed. The facility is roughly 36' long, 13' wide, and 8' high. This size is sufficient to contain the interior volume of any one ISS module. The facility consists of a 3-DOF (X,Y,Z) bridge-crane-like mechanism that supports a passive gimble that mounts the PSA, which permits free spinning in yaw and pitch. Currently, the facility supports 5-DOF motion (X, Y, Z, yaw, pitch). A gimbal, which permits yaw, pitch, and roll motion, will be mounted in the facility enabling 6-DOF motion in the near future. The bridge moves up and down the length of the facility. The trolley moves along the bridge permitting the trolley to move to any (X,Y) coordinate in the facility. A crane on the trolley raises and lowers the gimbal-mounted PSA. attached to it. The object to be tested is mounted in the gimbal and balanced so that it freely spins and doesn't "wobble." The micro-gravity test facility can be operated in the following four modes. The PSA Model 2 is shown in the facility in figure 3.

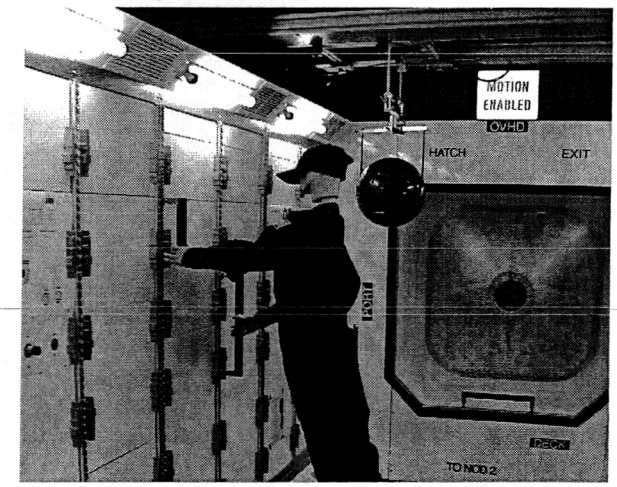

Figure 3 - PSA Model 2 in Micro-gravity Test Facility

The facility can be operated in several modes. The most significant mode enables us to simulate micro-gravity. Sensors located on the trolley and gimbal sense translation forces (X,Y,Z) acting on or generated by the gimbal payload. These sensor signals are interpreted by the crane motors as force commands and move the payload accordingly. The Z-axis signal is offset to cancel the force of gravity. The result is that an impulse force acting on the payload will cause it to "float" within the facility at a constant velocity. When the PSA Model 2 is the payload, its fan power is sufficient to propel it throughout the facility as if it was in a micro-gravity environment.

PSA Model 3and beyond

While testing continues on the Model 2, the preliminary design of the Model 3 is nearing completion. One notable difference between the Models 2 and 3 is the use of two blowers and four reactions wheels for propulsion and attitude control in the Model 3. The two counter-rotating blowers, located at the top and bottom of the sphere, exhaust through actuated vents to propel the PSA and the reaction wheels control its orientation. Although it is possible to control yaw, pitch, and roll with only three reaction wheels, a fourth reaction wheel enables momentum to be shifted among the reaction wheels. Another difference in the Model 3 is that it will include additional environmental sensors, including a thermal imager. Though when completed the Model 3 will be oversized and not space qualified, it otherwise will have all of the capabilities planned for the flight model.

The Model 3 design is a point design midway between

the Model 2 and the 8" dia. Model 4. A mockup of the Model 4 is shown in Figure 4. The 12" Model 3 will be similar in appearance and functionality to the Model 4, but will avoid the development of custom integrated circuits and other components required for the Model 4.

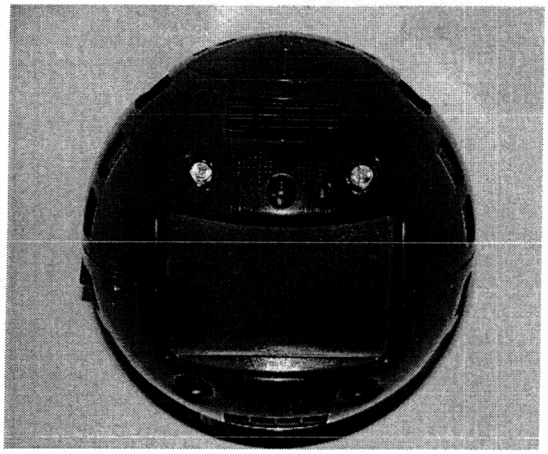

Figure 4 - PSA Model 4 Concept Model Mockup

Simulators

A variety of software simulators have served a crucial role in the software development process. They permit unit testing of components being developed as well as system integration tests when software changes are made. Our primary simulator is configurable so that it can replace various hardware and software components as needed for testing.

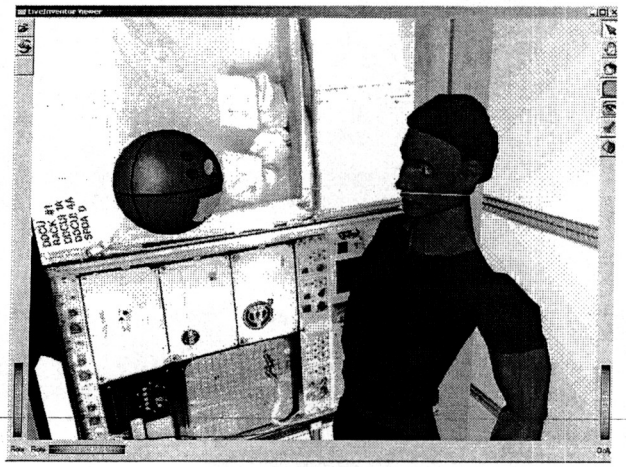

Figure 5 - 3D simulator screenshot of PSA in ISS with crewmember

We have recently integrated our PSA-specific simulator with a general-purpose 3D simulator, which provides 3D-rendered graphics and object dynamics. The current version is a synthesis of the graphics provided by the SGI Open Inventor™ 3D toolkit built on top of Open GL® and the CMLabs Vortex rigid-body physics simulator. By providing VRML and collision models of the ISS and objects within it including a PSA, we can navigate the PSA throughout the ISS and interact with simulated crewmembers, payloads, and objects that behave with realistic dynamics. The simulator is capable of supporting the simultaneous operation of multiple PSAs. In the Figure 5 simulator screenshot, a PSA is shown with a crewmember in the ISS U.S. Lab "Destiny" module.

In addition, this simulator has a scripting language that can be used to control the simulation. We have added an environment simulator to it to simulate fires, pressure leaks, and other faults to test the diagnostic capabilities of the PSA and its autonomous control system. We use the same autonomy software, often executing similar scenarios, to control the physical PSA prototypes in the physical simulators. These scenarios are helpful in testing the fidelity of the software simulators as well as the PSA hardware and software.

4. Autonomy Framework

An autonomy framework designed to address the previously discussed operational requirements has been developed and is depicted in Figure 6. The same software is used to command the PSA Model 1 and Model 2 as well as the PSA in simulation. Care was taken to design and implement this framework so that it is applicable to a wide range of free-flying vehicles.

The user can issue commands to the PSA through the Crew GUI. Also, the user can issue verbal commands to and receive spoken notifications generated by the PSA via a headset. Other external systems, including other PSA's, can directly and simultaneously issue commands to the PSA, which will attempt to resolve any conflicts. Finally, the PSA itself can generate commands in keeping with its high-level goals and periodic task schedule.

The PSA autonomy framework is comprised of a number of control elements, which are represented as boxes in Figure 6. The current implementation is distributed over three processors, as indicated by the dashed boxes, which are connected by wireless Ethernet. Each of these three subsystems and the control elements it contains is briefly discussed below. Note that the framework design and many of its elements draw their heritage from the model-based, goal-achieving, temporally-flexible NASA "Remote Agent" autonomy software flight-validated on the Deep Space One spacecraft in 1999 [2].

Onboard Control System Elements

The onboard control system is responsible for sensing, sensor analysis (e.g., object and fault recognition), state estimation (e.g., position estimation), hardware actuation (e.g., motor currents), and real-time reactive control (e.g., obstacle avoidance), generally with sub-second latency. This system is designed to enable local operation of the PSA even when communication with the off-board system is lost, which may occur during a flight emergency.

Local Path Planner—generates a trajectory between two waypoints that takes into account locally sensed obstacles When given a third waypoint, the trajectory passes through the second waypoint. The local path planner performs limited trajectory repair in case of a path plan failure, e.g., blocked path.

High-level controllers—translates the trajectory into a sequence of 6-DOF (position, velocity, and acceleration) setpoints for the low-level controllers.

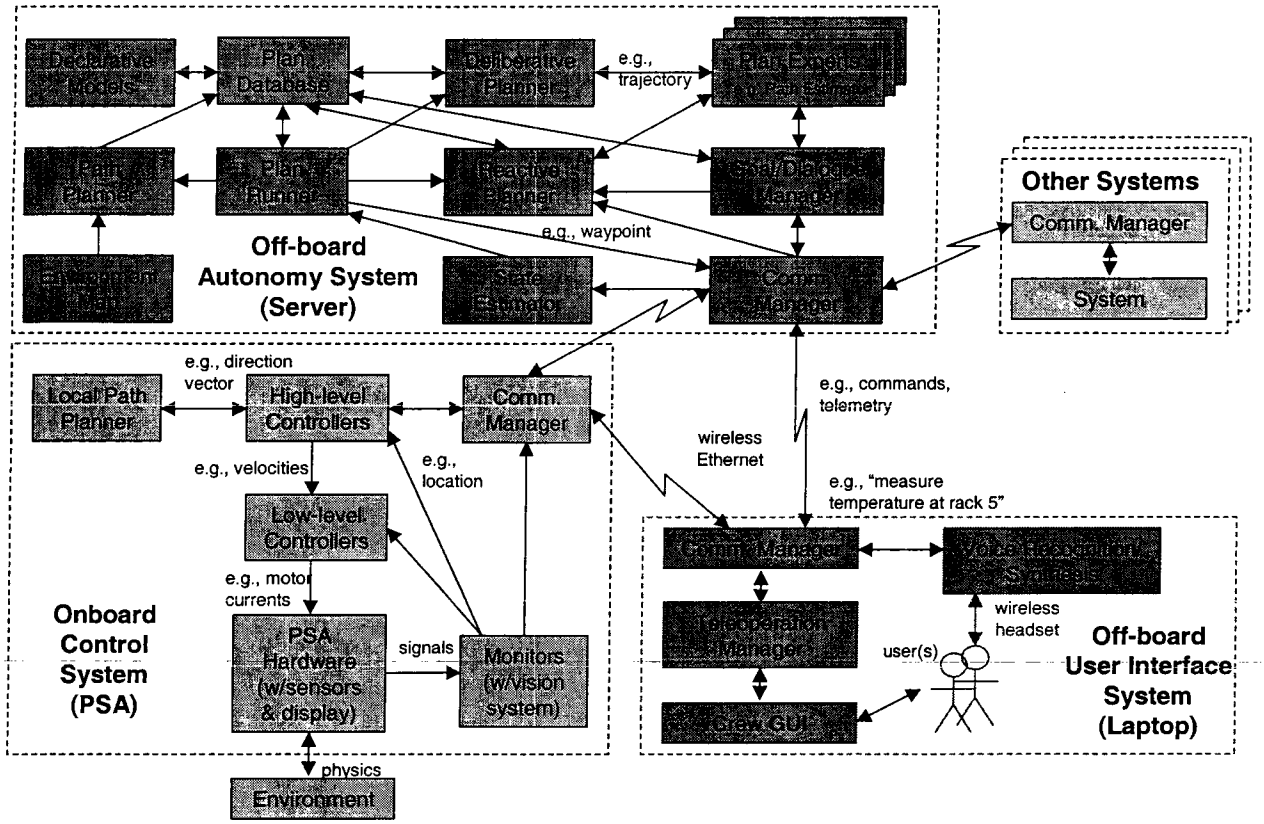

Figure 6 - PSA Autonomy Framework

Low-level controllers—translates the setpoints into motor force commands to achieve the specified PSA motion.

PSA Hardware—the sensors and actuators with their associated drivers. These include fan motor controllers, stereo cameras, environment sensors, proximity sensors, and an LCD.

Monitors—signal processing loops that abstract the data generated by the sensors. They run from being as simple as indicating that a proximity sensor has triggered to continually calculating 6-DOF positions and velocities by fusing the stereo camera, 6-DOF inertial sensors, and proximity sensors.

Communication Manager—responsible for managing message traffic and executing appropriate message handlers. Serves same role in both off-board systems.

Off-board Autonomy System

The off-board autonomy system is responsible for high-level autonomous control including inter-agent communication and coordination (including humans), goal management, decomposing high-level tasks (planning) into commands that can be executed by the onboard control system, e.g., waypoint commands, constraining task times (scheduling), command sequencing (plan execution), and reasoning about sensor data provided by the onboard control system, e.g., for diagnosis, and for plan repair, e.g., onboard control system is unable to achieve a waypoint. Architecturally, this system could be integrated onboard the PSA. It is off-board to permit increased computational power that is not constrained by onboard size, power, and communication constraints. The off-board processor also can be conveniently located in the PSA docking bay.

Declarative Models—contains the library of constraints used by the Plan Database that define a set of coordinated state machines. A constraint may simply specify that Task A must precede Task B by at least 10 seconds but not more than 20 seconds. The constraint may also functionally relate the parameters of tasks A and B as well as specify preconditions as to when it applies.

Plan Database—contains the plan being executed and is responsible for automated sub-goaling of tasks, i.e., determining the set of sub-tasks required to achieve a task, and for maintaining flexible plans, i.e., the propagation of valid task variable domains that are minimally restricted without violating a constraint. This has been implemented using the EUROPA plan database developed at the NASA Ames Research Center. EUROPA is a derivative of the model-based, temporally-flexible Remote Agent Plan Database described in [3], an earlier version of which was demonstrated on Deep Space One [2]. The plan database represents a temporal, constraint-based network of tokens that defines the past, the present, and flexibly-defined future states and actions of the system. Each token represents the "state" of a state variable for a period of time and the tasks that achieve or determine this state. Each token defines a start, end, and duration temporal variable, each with an upper and lower bound, as well as the procedure (predicate and arguments) invoked when the token is "executed." The plan database supports multiple

timelines with constraints on and between tokens. If none of the constraints are violated for a given instantiation of the plan database, the database is defined to be consistent.

Deliberative Planner—schedules outstanding tasks, and the related sub-tasks generated by the plan database, as well as makes decisions regarding constraining the domains of task variables to achieve specified goals during a specified period of time. This element is implemented by a variation of the Remote Agent Model-based Planner/Scheduler described in [3] and as specified by the Intelligent Distributed Execution Agent (IDEA) architecture [4]. More specifically, the Deliberative Planner (DP) is responsible for generating a consistent, flexible plan in the plan database given a start and end horizon time bound, an initial state of the timelines at the start time, and a set of goals. A flexible plan is loosely defined as a set of timelines, each consisting of tokens on each timeline, token order constraints that prevent overlapping tokens on the same timeline, and token procedure variable constraints. Plan flexibility is characterized by the set of decisions yet to be made that result in a consistent plan. A plan identification function is used to determine which of the outstanding decisions must be made in order to have a valid plan. The search process and decision selection priorities are determined in part by user-defined heuristics. Complex plans can require considerable computation time. The proper set of heuristics can dramatically reduce the time required. The DP is called to initialize the plan database and also is called during plan execution as specified by the plan being executed. It is typically called to plan for a period of significant duration sufficiently in the future such that the DP will complete prior to the start time of this period, but not so far in the future that the initial state at the future start horizon is not known with high confidence.

Reactive Planner—responsible for insuring that the Plan Database is in a state such that the tasks to be executed at a specified time are unambiguous. It has been implemented as described in [4]. In many respects, as implemented the Reactive Planner (RP) is very similar to the DP described above, although that not need be the case. The salient differences between the two planners are:

- the RP reasons over a shorter, more immediate time horizon, typically ending just after the current execution time
- the RP plan identification function is more restrictive so decisions that were postponed by the DP must now be made; the time allocated for planning is relatively very short, typically less than a few seconds, and cannot be exceeded without a fault
- in the event of a plan deliberation or execution failure, the RP repairs the plan locally or if necessary generates a standby plan to safe the PSA and calling the DP. Plan repair may be necessary for several reasons including tasks completing too late or too early, task return state variables posted to the Plan Database make it inconsistent, and new tasks have been added to the Plan Database for immediate execution that cause a conflict.

Plan Experts—computational procedures, called by a planner, that return information used by the planner to make planning decisions, typically regarding token variable values. For example, a route planner expert is called by either the deliberative or reactive planner to determine the time, route, and energy required to move between two points in the environment or to cover a certain space. The route planner expert has access to a global map that can be updated with sensed obstacles. A route plan request is typically made by the deliberative planner as part of developing the initial plan, but may also be called by the reactive planner to develop an alternate route if necessary, e.g., the route is blocked or there is insufficient energy to complete the current plan. In addition, a user may initiate a request to answer a hypothetical question about a particular goal.

Plan Runner (command sequencer)—executes tokens in the plan database at the appropriate time. Executing a token involves calling the procedure with its arguments defined by the token, updating the plan database with the token return values when the procedure terminates, constraining the plan database so that planners only have limited ability to change the past, and calling the Reactive Planner, as described above, as needed to update the plan database. The plan runner implemented is described in more depth in [4].

State Estimator—abstracts and infers a consistent set of state variables with respect to a system model given the discrete and continuous sensor data provided over time. Some of these state variables, such as the health of a sensor, may not be directly measurable. To accomplish this we are using the model-based L2 state estimation system, which is based on algorithms described in [5] and is an extension of the Livingstone system that was a component of the Remote Agent [2]. In certain instances it may be necessary to infer that a sensor is not healthy in order to achieve a set of state values that are consistent with the system model. In other cases it may be necessary to collect additional data to disambiguate between conflicting possible inferences for given sensor data.

Goal/Dialogue Manager—acts as an arbiter between the autonomous control system and other agents, including people. It retains state regarding its interaction with the other agents, e.g., recalls the subject of a previous sentence spoken by the user. As an arbiter, this element serves two roles: a goal manager and a dialogue manager. The goal manager essentially acts as a meta-planner for the deliberative planner. As stated above, the deliberative planner requires a start and end horizon time bounds, an initial state of the timelines at the start time, and a set of goals. The goal manager interacts with the user to determine this information. This may include negotiation of goals when all goals are not achievable or supporting mixed-initiative planning for hypothetical situations. The dialogue manager is responsible for acting as an intelligent interface with other agents. When interacting with people, it can converse with a person speaking a restricted natural language, responding as appropriate to spoken commands and queries. It inserts, changes or removes tokens in the Plan Database or responds to user queries by querying the planner experts and Plan Database. Currently, the integrated Dialogue Manager is simplistic. A more sophisticated dialogue manager tested on a stand-alone simulator is presented in [6]. The integration of such a dialogue manager remains as future work.

Step	Agent	Scenario Step Description
1	ECLSS	Detects a high heat signal from a fixed ISS node sensor. Fixed sensor health or heat source unknown.
2	ECLSS	Commands PSA to verify the temperature at that location
3	PSA	Generates plan upon receipt of the command to go to the fixed sensor location and measure temperature
4	PSA	Starts executing plan
5	PSA	Moves to fixed sensor and begins collecting temperature data and sending it to ECLSS
		Variation A: Fixed rack sensor failed high, rack lockers nominal
6a	ECLSS	Determines fixed sensor is faulty and uses PSA sensor as temporary sensor.
7a	ECLSS	Requests crewmember to repair sensor
8a	Crewmember	Repairs fixed sensor and notifies ECLSS
9a	ECLSS	Requests PSA to measure temperature to validate fixed sensor, which signals actual temperature
10a	ECLSS	ECLSS infers problem resolved and commands PSA to return to docking bay
11a	PSA	Returns to docking locker recharge
		Variation B: Fixed rack sensor healthy, one rack locker overheating
6b	ECLSS	ECLISS determines fixed sensor is accurate.
7b	ECLSS	Commands PSA to locate heat source.
8b	PSA	Searches region for heat source and determines maximum heat is at location of locker X.
9b	PSA	Sends locker location and its temperature to ECLSS.
10b	ECLSS	Determines locker can be powered down and turns off power to locker. Temperature declines.
11b	ECLSS	Requests PSA to verify temperature has declined
12b	PSA	Sends locker temperature to ECLSS
13b	ECLSS	Releases PSA to perform previously scheduled tasks
14b	PSA	Returns to docking locker to recharge

Figure 7 – Sample PSA Test Mission Scenario

Off-board User Interface System

The user-interface system enables the user to interact with the PSA by commanding and displaying information. It provides situational awareness, sensor-data views, plan views, and commanding capabilities. This includes interfaces for interactively creating and modifying the plan as well as teleoperation. Our intent is for this interface to support operation at various autonomy levels that can be dynamically changed and range from teleoperation to high-level autonomous control.

Voice Recognition and Synthesis—provides speech-to-text and text-to-speech conversions. The voice recognition subsystem essentially converts an audio signal into a parsed text stream. Conversely, the voice synthesis subsystem essentially converts text commanded by the Dialogue Manager or the Plan Runner into speech via the user headset or remote speakers. We use commercial products to accomplish these tasks and plan to upgrade them as improvements are made.

Teleoperation Manager—executes supported user commands and coverts GUI-generated commands into commands executable by the Autonomy system, e.g., plan editing. Also, it supports two force-feedback 3-DOF joysticks or one 6-DOF joystick for teleoperation in position, velocity, or acceleration modes.

Crew GUI—displays the sensor data, renders the PSA in a 3D model of its environment, displays plans, provides plan editors for both PSA task and path plans, and provides for direct commanding of the PSA. Included in the displayed sensor data is the real-time video stream generated by the PSA. In addition, by using a camera mounted on the Crew GUI display, the Crew GUI supports teleconferencing.

5. Sample Mission Test Scenario

In order to measure the system capabilities with respect to the operational requirements and to identify the challenging problems, several scenarios have been developed. These scenarios are designed to execute in simulation as well as with the physical prototypes in the test facilities. These scenarios perform a valuable role in measuring our current capability levels and are also useful for regression testing. As the capabilities of the PSA including its autonomous control system improve, the complexity scenarios are increased effectively raising the performance bar.

In this section we discuss a scenario, summarized in Figure 7, in which the PSA, an autonomous Environmental Control Life Support System agent, ECLSS, and a crewmember participate in the diagnosis of and recovery from an ISS module fault. In this scenario, ECLSS is autonomously controlled by a high-level autonomous system similar to the one used by the PSA as shown in Figure 6 (the main difference is that ECLSS does not use a path planner). The scenario has two variations depending on the cause of the initially sensed anomaly. This scenario is used to demonstrate:

- Integrated Vehicle Health Management
- Cooperative multi-agent planning and execution
- Generation and execution of a near-optimal 6-DOF route plans
- Stereo vision-based 6-DOF localization and map registration

The scenario begins with the PSA station keeping at its dock when a fixed temperature sensor at rack 5, locker 1 in the ISS U.S. Lab module signals a high temperature to the ECLSS. The ECLSS attempts to diagnose the problem and is not able to determine whether the sensor is defective or if the station system is actually overheating without additional information. In our case, we have specified that each case is equally likely. So in order to disambiguate the system state, ECLSS commands the PSA agent to go to the fixed sensor location and verify the temperature at that location by sending the PSA agent a sense-at-location goal. The PSA agent then reactively deliberates (i.e., the reactive planner calls the deliberative planner in response to the new goal). The deliberative planner decomposes the goal into a move-to subgoal

followed by a subgoal to maintain position while the temperature is sensed. The move-to subgoal then decomposes into a path-planning subgoal followed by an execute path subgoal. All of these goals are flexibly scheduled. When the path-planning goal is executed, a path from the current location to the desired location is generated, where a path consists of a sequence of waypoints that avoids known obstacles and no-fly zones. When the path is scheduled to execute, the PSA agent sends it to the PSA subsystem, which executes each waypoint. As needed, the trajectory between waypoints is dynamically changed to avoid obstacles detected en route. When it arrives at the destination, the PSA subsystem confirms that the path was completed, or in a failure case it cannot be achieved, with the PSA agent. PSA agent then commands the PSA subsystem to station keep for a period while the PSA measures the temperature. After that period, the top-level PSA agent sense-at-location goal completes by returning the sensed temperature to ECLSS. ECLSS then compares the two sensor readings. The two cases where they agree or disagree are listed below. The preceding activity is summarized by steps 1-5 in Figure 7.

If the PSA and ECLSS temperature sensors disagree, the ECLSS state estimator infers that the fixed temperature sensor has failed and requests that a crewmember repair it by sending a message to the ECLSS operator user interface requesting the repair and waits for confirmation that the crewmember has repaired the sensor. When the crewmember confirms, the fixed sensor value returns to nominal. Meanwhile, ECLSS tells PSA to measure the temperature again at the same location and compares the return value to the value read from the fixed sensor. Since they now agree, the ECLSS state estimator infers that the fixed sensor is healthy and the PSA is commanded to its dock completing the scenario (steps 6a-11a).

However, if the PSA and ECLSS temperature sensors agree, then the ECLSS state estimator infers that a nearby locker is overheating, but which one is unknown. ECLSS gives a goal to the PSA agent to direct it to locate the source of the heat. The PSA agent decomposes this goal to send a command to the PSA subsystem that causes it to execute its heat source seeking behavior. This behavior has the PSA first spin fully around, scanning the environment with its thermal imager. Once the scan is complete, the PSA points to the largest magnitude heat source and moves toward it. When the PSA gets as close as it can to the heat source, the PSA agent goal returns the location and temperature measurement to ECLSS. In our case, the heat source is actually in a neighboring rack: rack 4, locker 3, which the ECLSS state estimator infers. ECLSS then commands the system operating in the locker to power-off, which reduces the (simulated) heat in the area. The ECLSS fixed sensor then reads a nominal temperature. ECLSS sends the PSA agent a goal to measure the temperature again to verify the temperature is nominal. The PSA measures the temperature and returns the temperature to ECLSS. ECLSS infers that the locker temperature is nominal and releases the PSA from further requests. The PSA then returns to its dock completing the scenario (steps 6b-14b).

6. Future Work

Future research and development efforts will focus on system-level active hybrid diagnosis, fleet operations (several PSAs working together to handle environmental problems) as well as autonomous operations with spacecraft command and control systems (instead of human commanding/teleoperating). Long-term functional upgrades may include adding effectors, e.g., arms, capable of control panel operation, payload maintenance, re-supply, and repair. Consider a mission where a spacecraft is in orbit unoccupied. A larger, 4-armed PSA could be used to monitor and maintain the flight worthiness of the spacecraft and reduce mission risk.

7. Summary

We presented the ongoing research and development effort to design the autonomous control software for an internal spacecraft autonomous mobile monitor, which is also applicable to a wide range of free-flying vehicles. We discussed the high-level functional requirements of the project followed by a description of the PSA prototypes of increasing complexity and fidelity, as well as the micro-gravity test facility, which allows us to fly the PSA prototypes on the ground as if they were onboard the ISS. The autonomy framework for intelligent flight vehicle control being developed and tested as part of this project was then presented and its components detailed. A sample mission scenario being used to test the prototypes and the autonomous control system was also outlined. We concluded with a brief discussion of the future work.

8. Acknowledgments

We gratefully acknowledge the contributions of the many talented people on this project including Kurt Konolige, Nicola Muscettola, Charles Neveu, Eric Poblenz, and Adam Sweet. In addition, we acknowledge the support provided by the NASA Cross-Enterprise Technology Development Program, the Computing, Information, and Technology Program, and the Engineering Complex Systems Program.

9. References

[1] Gregory A. Dorais and Yuri Gawdiak, "The Personal Satellite Assistant: an internal spacecraft mobile monitor." *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, 2003.

[2] Douglas Bernard, et al., "Final report on the Remote Agent experiment." *Proceedings of the New Millennium Program DS-1 Technology Validation Symposium*, Pasadena, CA, February 8-9, 2000.

[3] Ari K. Jonsson, et al., "Planning in interplanetary space: theory and practice." *Proceedings of the 5th Artificial Intelligence Planning and Scheduling Conference*, Brekenridge, CO, 2000.

[4] Nicola Muscettola et el., "A unified approach to model-based planning and execution." *Proceedings of the Sixth International Conference on Intelligent Autonomous Systems*, Venice, Italy, 2000.

[5] James Kurien and P. Pandurang Nayak, "Back to the future with consistency-based trajectory tracking." Proceedings of the 17th National Conference on Artificial Intelligence, Austin, TX, 2000.

[6] Manny Rayner, Beth Ann Hockey, and Frankie James, "A compact architecture for dialogue management based on scripts and meta-outputs." *Proceedings of Applied Natural Language Processing (ANLP)*, 2000.