The NASA/Army Autonomous Rotorcraft Project

M. Whalley
mwhalley@mail.arc.nasa.gov
Army/NASA Rotorcraft Division
Aeroflightdynamics Directorate (AMRDEC)
US Army Aviation and Missile Command
Ames Research Center, CA

M. Freed
mfreed@mail.arc.nasa.gov
Institute for Human and Machine Cognition
West Florida University
Computational Sciences Division
NASA Ames Research Center, CA

M. Takahashi
mtakahashi@mail.arc.nasa.gov
QSS, Army/NASA Rotorcraft Division
NASA Ames Research Center, CA

D. Christian and A. Patterson-Hine
dchristian@mail.arc.nasa.gov
apatterson-hine@mail.arc.nasa.gov
Computational Sciences Division
NASA Ames Research Center, CA

G. Schulein
gschulein@mail.arc.nasa.gov
San Jose State Foundation
Army/NASA Rotorcraft Division
NASA Ames Research Center, CA

R. Harris
rharris@mail.arc.nasa.gov
QSS, Computational Sciences Division
NASA Ames Research Center, CA

An overview of the NASA Ames Research Center Autonomous Rotorcraft Project (ARP) is presented. The project brings together several technologies to address NASA and US Army autonomous vehicle needs, including a reactive planner for mission planning and execution, control system design incorporating a detailed understanding of the platform dynamics, and health monitoring and diagnostics. A candidate reconnaissance and surveillance mission is described. The autonomous agent architecture and its application to the candidate mission are presented. Details of the vehicle hardware and software development are provided.

INTRODUCTION

Enabling an unmanned helicopter to execute fully autonomous low-altitude scientific or military missions requires technologies that are complex and largely unrealized. For example, just to get to where it needs to go, an autonomous helicopter would need to sense, classify, and identify landmarks, reconcile those landmarks with stored maps, localize itself to those landmarks, rapidly compute a path that would keep it away from perceived obstacles or threats, and closely follow that path in the presence of environmental disturbances. And all of this is before beginning to address the decision process underlying the prosecution of mission objectives. With few exceptions, these topics remain as unmet research challenges. Due to the lack of available autonomous technologies, unmanned vehicles in operational use require intensive operator oversight and control for even the simplest missions. Both NASA and the US Army seek to advance unmanned vehicle operations beyond low-level control and significantly increase mission complexity and capability.

Presented at the American Helicopter Society 59th Annual Forum, Phoenix, Arizona, May 6–8, 2003. *Copyright © 2003 by the American Helicopter Society International, Inc. All rights reserved.*

NASA desires to free itself of the arduous task of having mission controllers preplan every minute action of planetary explorers. A rover crawling a few meters per day across the surface of Mars, based on painstaking path planning by Earth-bound experts, reduces scientific discovery to a trickle. Far greater science output could be achieved if missions could capitalize on the range capabilities of a self-directed rover, submarine, or helicopter.

The US Army seeks to exploit the low-altitude, hovering, and vertical takeoff and landing capability of unmanned helicopters without the intensive planning and execution required to avoid obstacles and threats. The mission flexibility and situational awareness required by Army applications will demand much more than can be achieved by flying from waypoint to waypoint, high above any obstacles, dependent upon clear GPS signal reception. This means developing the autonomous technologies that can deal with the hazards encountered in the low-Earth environment and enabling self-directed mission planning and execution with little or no operator input.

The Autonomous Rotorcraft Project (ARP) is developing an all-inclusive autonomous helicopter research platform using unique in-house skills in helicopter guidance and flight control, robotics planning and scheduling, and emerging UAV sensor technology. Unlike part-task research platforms, the comprehensive ARP autonomous helicopter system will be capable of identifying and addressing those weaknesses that most impact mission effectiveness. This will be done so that

ARP can provide the needed design guidance to future NASA and Army system development efforts.

The remainder of this paper will describe ARP efforts to identify and solve the challenges of helicopter autonomy. Included in this description is a candidate mission on which ARP development is focused, the associated vehicle hardware and software development, and the chosen autonomous agent architecture and its application.

PROJECT MOTIVATION

ARP is motivated by needs formally originating within the NASA Computing, Information, and Communications Technology Program (CICT, Ref. 1), the NASA Bio-inspired Engineering for Exploration Systems Project (BEES, Refs. 2 and 3), and the US Army Precision Autonomous Landing Adaptive Control Experiment (PALACE, Ref. 4).

The NASA CICT program was established in 2001 to ensure NASA's continuing leadership in developing and deploying key enabling technologies for a broad range of mission-critical tasks. ARP is primarily motivated by and funded under the Intelligent Systems (IS) portion of CICT, a key objective of which is to develop systems that make decisions with limited intervention. To address this objective, ARP intends to develop, demonstrate, and assess the capabilities of automated reasoning technologies in the context of the complex rotorcraft environment. A rotorcraft serves as an ideal platform for developing and demonstrating automated reasoning software for Mars landers, aircraft or satellite clusters, and other NASA flight applications. The complex high-bandwidth dynamics and cluttered, unpredictable operational environment provide an excellent surrogate for the kinds of challenges likely to be faced in a remote robotic explorer mission.

The vision of the BEES program is that small, mechanical platforms which mimic the mobility of biological systems, can be built at low cost, instrumented and used as platforms for carrying scientific instruments. ARP will support the BEES program by providing a platform on which to flight test surface feature recognition sensor technologies currently under development (Ref. 5).

The US Army plans to address a broad range of vertical takeoff and landing (VTOL) UAV topics in the coming years. Under the PALACE Science and Technology Objective (STO) it has identified accurate, reliable autonomous landing at remote un-instrumented sites as a challenging and critical capability. ARP will contribute to meeting the goals of the PALACE STO by providing a platform for the demonstration and integration of the various technologies. This will include developing and flight demonstrating autonomous landing on a slope, in moderate wind, and in the presence of obstacles.

PROJECT GOALS

To address the above objectives, ARP has adopted the following project goals: 1) close integration of a reactive planner with navigation, flight control, and mission systems, 2) aggressive maneuvering in a real-world environment including obstacle avoidance and landing at unprepared sites, 3) incorporation of vehicle health into mission planning, and 4) mission planning rapid enough to cope with the high-bandwidth dynamic characteristics of small-scale helicopters.

RECONNAISSANCE AND SURVEILLANCE MISSION DEFINITION

Current development efforts are focused on creating a flexible autonomous reconnaissance and surveillance (R&S) capability. Fundamentally, this mission requires that the UAV helicopter act with the goal of maximizing the value of returned information. In pursuing this goal the vehicle will need to balance various activities such as patrolling established (pre-optimized) routes, dynamically modifying routes in response to evolving situation information or to avoid threats, investigating targets in response to special contingencies, maneuvering to obtain optimum sensor perspectives, repositioning to transmit gathered information, and periodically returning to base to refuel. Furthermore, the system must be capable of seamlessly integrating information provided by external systems or the desires of human users who may wish to influence mission prioritization or conduct.

Mission flexibility will be achieved by allowing the overall R&S information gathering requirements to be expressed without specifying a particular plan of action (e.g., “go to the warehouse; then orient camera one on the entrance; then take a photo; then go to the loading area; etc.”). Instead, the user will define general preferences, plan knowledge, and target or terrain characteristics (see Ref. 6 for a similar approach). For example, it may be specified that a perimeter fence line or building should be checked periodically for signs of intruders, that it would take approximately t minutes for an intruder to carry out an undesirable action, that intrusion attempts will tend to occur at frequency f, and that there is an expected cost c for failing to detect an intruder. Such a set of parameters would result in a functional description of the importance of visiting a target that is nonlinear with respect to time. For instance, there may be a period immediately following an observation where the importance of revisiting a target to check for intruders remains low and unchanging. This may be followed by a period where the importance increases rapidly. Finally, this would be followed by a period where it decreases to zero because in all probability the damage has been done and the intruders have long since escaped. On the basis of such knowledge, autonomy mechanisms would reason in a decision-theoretic sense about how best to maintain surveillance over a set of targets including optimizing target sequencing, determining safe routes, choosing the proper sensor, maneuvering for the best camera angles, and determining minimum dwell time.

These same autonomy mechanisms will make it possible to adapt surveillance decisions in response to events that may be difficult for a person to anticipate or respond to rapidly. For example, on becoming informed that some friendly entity has just examined one of its surveillance targets, the UAV might delay subsequent surveillance of that target. This reflects an ability to balance the goal of maintaining current (non-obsolete) information about a target against the opportunity cost of examining one target instead of another. In contrast, intruder alarms sounding at several targets at once would require rapidly modifying surveillance strategy in a dramatic way to reflect an increase in urgency for examining those targets.

The R&S mission provides a focus that balances the desire to demonstrate a flexible autonomous system against the computational and sensor capabilities of the platform. The remainder of this paper will describe the hardware and software development efforts to support such a demonstration.

HARDWARE DEVELOPMENT

Two Yamaha RMAX helicopters are used as demonstration platforms for ARP (Fig. 1). The RMAX was originally developed for remote control agricultural seeding and spraying but has been adapted here for use as an autonomy demonstration platform. The aircraft is capable of approximately one hour of hover flight duration with a 65 lb payload. The payload capability makes it possible to use off-the-shelf sensor and computer hardware and avoid the cost and complexity of component miniaturization.

Fig. 1. ARP RMAX research aircraft.

An avionics payload and stub wing camera mount have been developed for use with the ARP RMAXs. Total weight of the payload and wing is approximately 45 pounds. The payload is shown in Fig. 2 and its components are listed in Table 1. Two computers are used to distribute the computational load and to separate the more computationally intensive vision processing from the critical flight control tasks. The payload is powered by the RMAX generator, which has been oversized to provide approximately 100 watts of power to the research hardware. The carrier-wave-phase-tracking differential GPS system provides centimeter level accuracy relative to the base station. The three radio modems function as a single unit and are collectively capable of 345k baud transmission rate but typically sustain approximately 200k baud. With a maximum transmission power of one watt, the radio modems are theoretically capable of a 20 mile range, but this has not been tested.

The avionics payload PC/104+ computer communicates with the RMAX on-board Yamaha Attitude Control System (YACS) computer via four serial lines. Three of the serial lines enable reading of the Yamaha rate and attitude sensor package, the standard RC-control radio receiver signals, and the Yamaha YACS computer internal variables. The fourth serial line serves to bypass the actuator commands generated by the Yamaha YACS computer thus enabling the avionics payload to serve as the flight control computer. Communication transport delays of approximately 40 msec have been measured for each of the serial lines significantly impacting flight control law performance.

Engagement of the PC/104+ computer as the flight computer is achieved via a pushbutton on the Yamaha RC transmitter. A watchdog timer on the Yamaha YACS computer causes control to revert to the standard RC receiver when actuator commands stop being received on the serial line. A second timer causes the helicopter to enter a full-down collective and throttle idle setting if there are neither RC nor serial line actuator commands being received.

Mounted externally is a vibration-isolated stub wing with four digital cameras. The cameras are shown in Fig. 3 and the wing components are listed in Table 1. The Unibrain camera will serve initially as a situational awareness aid. The stereo pair of cameras have a 40-inch baseline and are intended to provide input to a passive range estimation algorithm on the compact Peripheral Component Interconnect (PCI) computer. The camcorder serves as a full-rate on-board video recording mechanism. All four cameras are connected via a Firewire hub to the compact PCI computer and are fully controllable via that link.

An instrumentation trailer has been developed to support development and testing activities (Fig. 4). The trailer contains extensive resources including two Linux-based workstations and one Mac G4 workstation for use in development and operation. An on-board Ethernet switch and satellite hubs provides easy expansion capability for laptops. Each workstation is equipped with dual flat-panel LCD displays. There are also a GPS ground station and three radio modems for communication with the aircraft. A video distribution system enables display of vehicle situational awareness information on the upper display at each workstation.

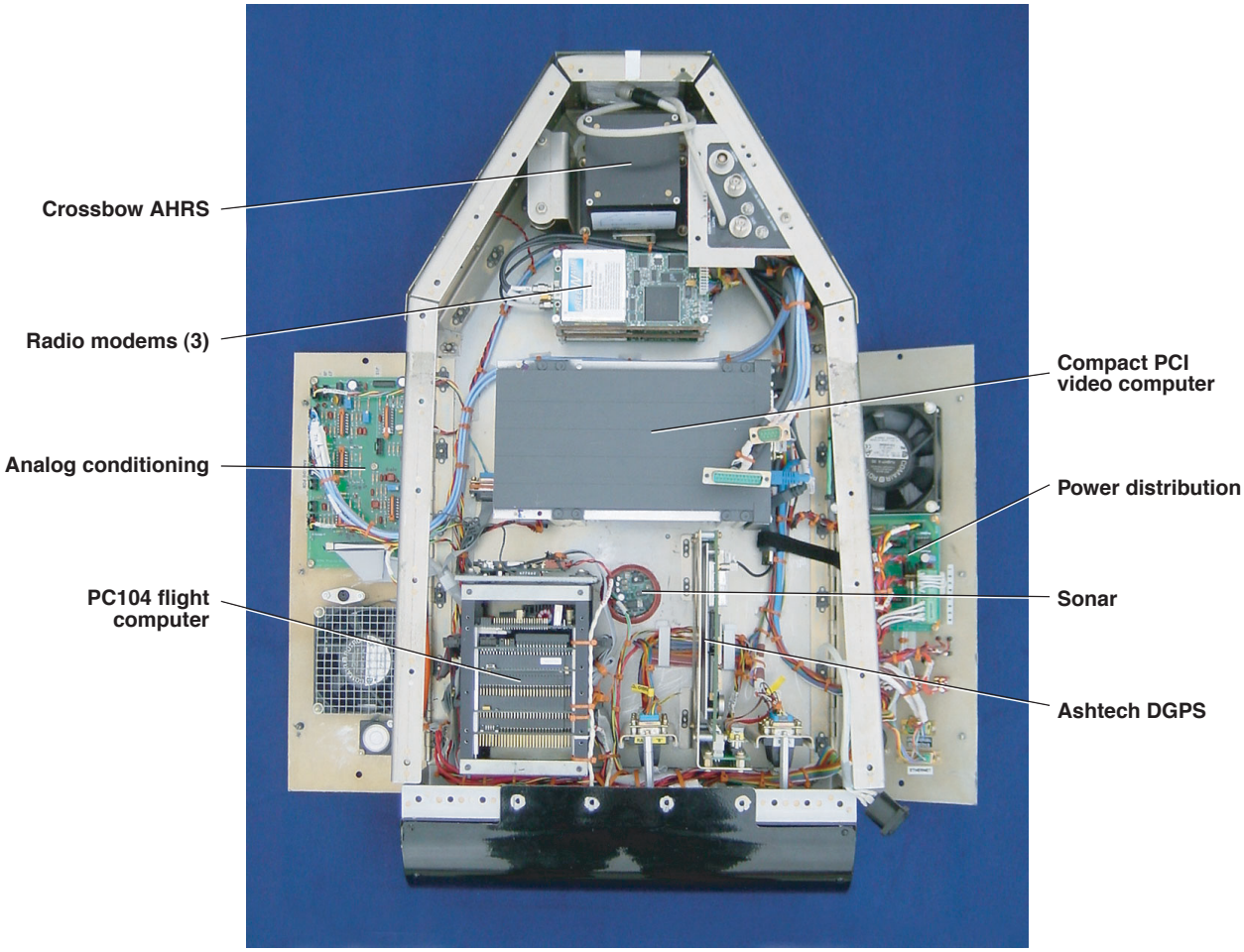

Fig. 2. Payload components (lid removed and side doors opened for clarity).

Fig. 3. Point Grey digital camera and Canon camcorder (left); Unibrain and Point Grey digital cameras (right).

Fig. 4. Instrumentation trailer and aircraft; one of four workstations (inset).

Table 1. Avionics payload and wing components.

Component	Description
Avionics Payload	
PC/104+ computer	Versalogic Panther processor (400 MHz AMD K6), Connect Tech Xtreme/104 8-port serial communications board (460.8 Kbps), RTD DM7520-8 ADC board, 512 Mbyte high-speed CompactFlash memory
Compact PCI computer	Inova ICP-PIII (700MHz Pentium III, fanless) with integrated Ethernet and IEEE-1394 interfaces, 3U cPCI chassis
Crossbow AHRS IMU	Rate, attitude, heading, and acceleration sensor employing MEMS-technology and three-axis magnetometer, RS-232 interface
Ashtech DGPS	Ashtech/Magellan Z-sensor GPS with Real Time Kinetic (RTK) and 10 Hz output options, RS-232 interface
900 MHz radio modem (2)	Freewave DGR09 radio modem, 115.2 Kbps, 1 watt (25 mile range with 3 dB antenna), RS-232 interface
2.4 GHz radio modem	Freewave I-Series radio modem, 115.2 Kbps, 500 mw (20 mile range with 3 dB antenna), RS-232 interface
Power distribution PCB	Project power distribution (custom-built)
Analog conditioning PCB	Analog conditioning board (custom-built)
Sonar	EDP Ultrasonic Sonar Transducer, 6 in to 10 ft range, analog interface
Accelerometers	Measurement specialties ACH-01, 150 g, low pass filtered at 5 kHz; used for vibration measurement at different locations
Temperature sensors	National Semiconductor LM60 solid state temperature sensors; used for temperature measurement at different locations
Stub Wing	
Unibrain Fire-i400 camera	640x480 pixel machine vision camera, C-mount lens, IEEE-1394 interface
Point Grey DragonFly camera (2)	640x480 pixel monochrome camera, stereo pair with 40-inch baseline for passive ranging, IEEE-1394 interface
Canon camcorder	DV camcorder with progressive scan mode, IEEE-1394 interface
Firewire hub	IEEE-1394 hub interconnects cameras and Compact PCI computer
Other	
Weight-on-wheels sensors	Sensotec Model 13 analog force transducers (250 lb range) with custom amplifier, transducers integrated into rubber backstops for skids
900 MHz antenna	MaxRad 900-928 MHz 3 dB antenna on stub wing (2); MaxRad 5 dB antenna on ground (2)
2.4 GHz antenna	Mobile Mark 5 dB antenna on tail boom; Mobile Mark 9 dB antenna with cardioid reflector on ground

SOFTWARE ARCHITECTURE

All ARP operational and development computer environments are Linux-based. To the extent possible, all software being used including flight control, image processing, communications, telemetry, and health monitoring has been developed in-house or adopted from open-source software. All project software is version controlled using Concurrent Version System (CVS) tracking.

The on-board and ground software architecture is shown in Fig. 5. It is important to note that the architecture has been specifically designed so that identical source code is used in the development environment, during hardware-in-the-loop testing, and in flight. This provides confidence in the integrity of software before field testing.

The elemental software modules are:

domsD – a Distributed Open Messaging system (DOMS) communications daemon which runs on each ARP computer. All communication between processes is achieved via the DOMS daemon (described in detail in a later section).

taskMaster – a process launcher and monitor that runs on each ARP computer. The taskMaster is responsible for starting ARP programs in a specified order as well as adding desired delays in the startup sequence. This module also monitors a heartbeat generated by each process it starts and then restarts them in case of unexpected termination.

CLAW – the inner and outer loop Control LAWs for the RMAX helicopter. This module implements an attitude-command/attitude-hold (ACAH) system and the path following system. This program also includes all engage and disengage logic and has a simple internal aircraft model for use during hardware-in-the-loop ground testing.

healthPlus – monitors various system health related parameters such as vibration and temperature. This program also flashes external LEDs that indicate the state of the Apex reactive planner.

GPS – reads the onboard GPS and communicates coordinates to other ARP processes. This module accepts differential corrections from a ground-based GPS and provides them to the onboard GPS for greater accuracy.

GPS Base – reads the ground GPS and sends differential corrections to the onboard GPS.

Apex – the reactive planner described below.

RDS – Remote Diagnostic Server health monitoring software (described below).

Path smoother – algorithm that accepts predefined waypoints and returns a smoothed path using a natural spline fit optimized to keep the path within specified corridor constraints and adding straight line segments between widely-separated waypoints. Example output of the path smoother is shown in Fig. 6.

stateRip – DOMS-to-shared memory translation program which writes position and orientation of the aircraft into RIPTIDE shared memory.

RIPTIDE – Real-time Interactive Prototype Technology Integration/Development Environment (RIPTIDE, Ref. 7). This is used extensively for both development and visualization. It is used as a real-time desktop environment for navigation and flight control law development, as well as testing of Apex (see below). It is also used as a visualization tool during flight to provide a high-fidelity simulated view of the aircraft and operating environment synthesized from telemetered aircraft state and position data (Fig. 7).

Moving Map –shows the location of the aircraft and provides a human interface to Apex and the path following system.

videoSend, videoReceive –controls the onboard cameras and sends camera images to other processes.

APEX REACTIVE PLANNER

ARP employs Apex, an autonomy architecture designed to operate in uncertain task environments like that of the R&S mission (Ref. 8). The core element of Apex is a reactive planning algorithm (Refs. 9 and 10) that selects actions based partly on a library of stored *partial* plans. Such planning algorithms are considered reactive, because decisions about the next course of action evolve as new decision-relevant information becomes available. For example, reconnaissance of a particular location might be delayed in response to hazardous weather conditions or, alternately, increased in urgency if weather conditions are likely to make the route hazardous later. Similarly, a decision regarding how to get to the location might be made (or changed) at any time in the course of carrying out the overall R&S plan based on changes in the probable locations of hazards and information opportunities.

Reactive versus Classical Planners

The Apex reactive planning approach contrasts sharply with that of classical planners, which not only select all actions in advance of execution, but do so by constructing new plans rather than retrieving stored plans. The two approaches represent a tradeoff. Classical approaches, used in many robotic applications, are generally sound and complete; i.e. able to create plans of guaranteed validity if any such plan is possible. However, they generally require comprehensive and detailed information about the environment in which the plan will be executed. Uncertainty arising from such things as unpredicted events, changes in task priorities, or failures to successfully execute an action, undermine a classical planner. This often makes it impossible to generate a plan of even plausible validity. Moreover, classical planning is computationally intensive and thus slow. Therefore, the drawbacks of the classical approach can make it incompatible with the requirements of a practical R&S capability. In contrast, reactive planners are appropriate for the R&S mission as they were invented specifically to be both fast and robust to uncertainty.

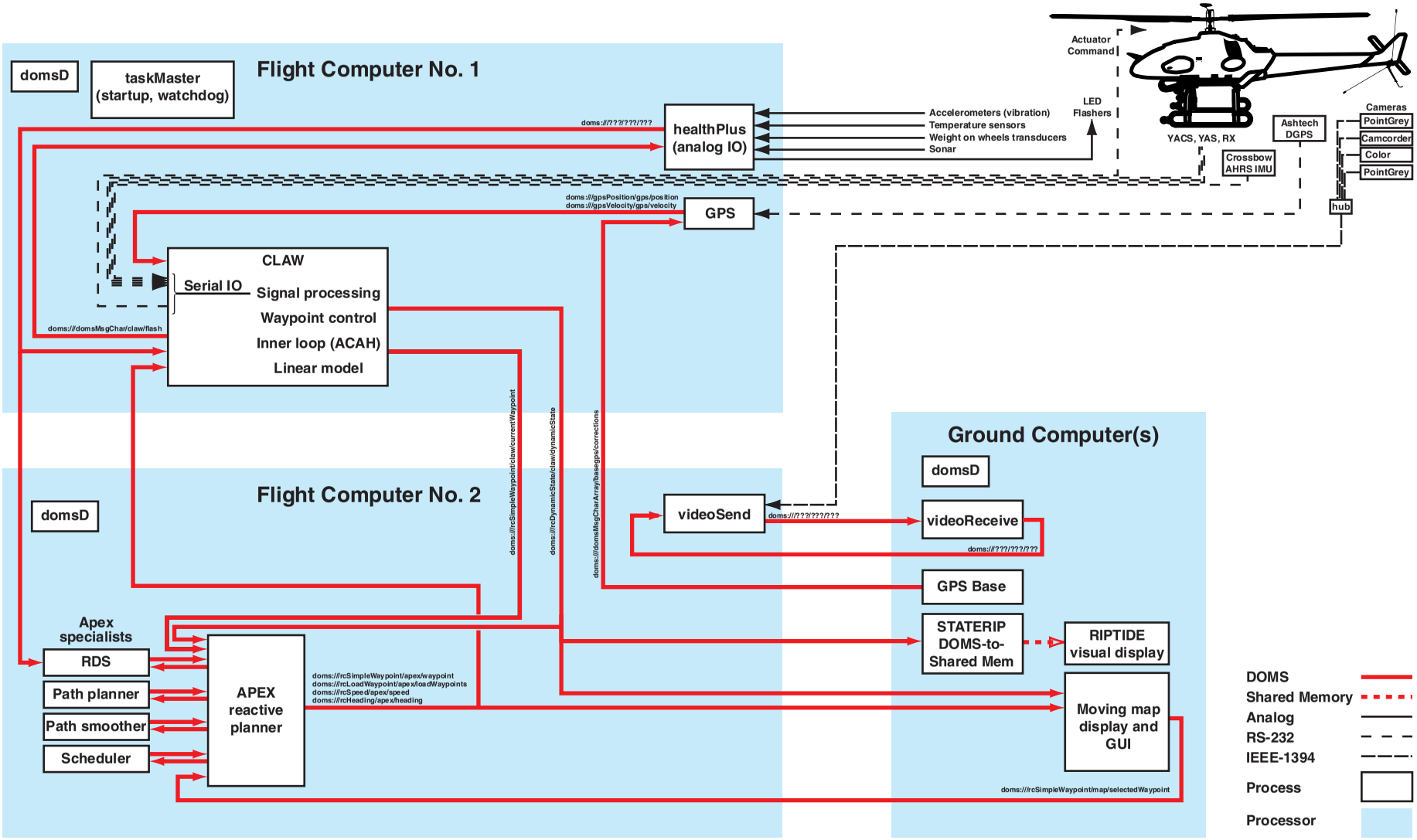

Fig. 5. Software Architecture.

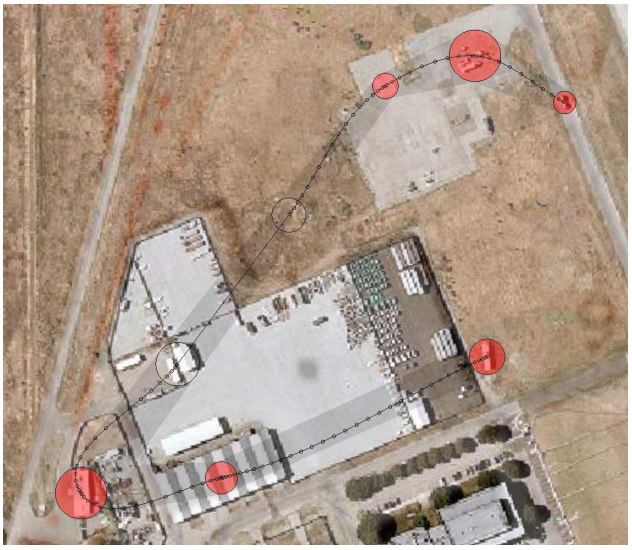

Fig. 6. Moving map showing output of path smoother; waypoints shown in red, constraint corridor shown in gray.

Fig. 7. RMAX and Instrumentation trailer (top); real-time RIPTIDE view of same (bottom).

The Apex architecture was originally designed to emulate human behavior in domains such as air traffic control where correct behavior is partially defined by a set of standard operating procedures. Standardizing procedures in such domains has several advantages that are believed to be applicable to R&S missions. These include enhanced ability to cooperate with other agents and greater optimization of behavior with respect to statistical regularities (risks and benefits) in the environment. Standardization also enhances predictability, which in turn facilitates human-system interaction. Carried out concurrently or interleaved, procedures can interact and potentially conflict. The main challenge in deciding action in such procedure-driven domains, either for an autonomous agent or a real human operator, is thus to coordinate the execution of multiple procedures.

Procedure Definition Language (PDL)

Apex synthesizes a course of action mainly by linking together elemental procedures expressed in Procedure Definition Language (PDL), a notation developed specifically for the Apex reactive planner. A PDL procedure consists of at least an **`index`** clause and one or more **`step`** clauses. The index uniquely identifies the procedure and describes a class of goals for which the procedure is intended. Each **`step`** clause describes a subtask or auxiliary activity prescribed by the procedure. Steps are not necessarily carried out in the order listed or even in a sequence. Instead, they are assumed to be concurrently executable unless otherwise specified. If a fixed step ordering is desired, a **`waitfor`** clause is used to specify that the completion of one step is a precondition for the start of another.

Fig. 8 shows a simple example of PDL. In the example, the step labeled *checkout* waits for the ascent to hover action to complete. The monitoring steps (labeled *mon1* and *mon2*) must wait for the checkout step to complete, but they are not otherwise constrained by the logic of the procedure. In particular, no ordering constraints are imposed between monitoring steps. If the procedure was allowed to control two UAVs, it would allow both monitoring steps to be executed in parallel. In this case, Apex would automatically detect that the two steps cannot be carried out concurrently and then attempt to resolve the conflict on the basis of preferences (such as those specified in each step's priority clause) and current situational information. Conflict resolution in such cases means deciding order and is thus inherently a scheduling problem. The set of conflicting tasks – there may be more than these two, since other procedures may be executing in parallel – are passed to Apex's priority-based scheduler, which attempts to optimally sequence the tasks.

The example above illustrates two ways order can be determined: 1) by explicit constraints in a stored procedure, or 2) by preference criteria (priorities) employed by scheduling mechanisms to resolve conflicts. This flexibility is at the heart of Apex's integrated approach, allowing the system to draw on the capabilities of a reactive planner or on those of a scheduler as appropriate. This integration of scheduling with reactive planning has proven crucial for past applications of the Apex framework (Ref. 11). The main elements of the current approach are described in Refs. 8 and 12. In future work, the system's scheduling capabilities will be extended to take advantage of a more sophisticated model of priority.

```
(procedure
   (index (do surveillance sector-1))
   (step start (ascend to hover (50 feet)))
   (step checkout (test telemetry) (waitfor ?start))
   (step mon1 (monitor warehouse-1 still-image) (waitfor ?checkout)
        (priority (theft) :urgency (30 min) :importance 100))
   (step mon2 (monitor north-entrance still-image) (waitfor ?checkout)
        (priority (intrusion) :urgency (20 min) :importance 40))
   (step aux1 (refuel at base) (waitfor (fuel-level low))
        (priority (vehicle-health) :urgency (5 min) :importance 200)))
```

Fig. 8. Sample Apex PDL.

User Input to Apex

The described approach is meant to support the primary goal of performing R&S without human intervention. However, it is anticipated that the system will interact with humans in several ways ranging from narrow interventions in runtime behavior (e.g., adding a new surveillance target) to creating new operating procedures for the vehicle to follow. The goal is to support interactions across this range, insuring that users will not have to learn and write PDL procedures for simple interactions, but also going as far as possible in supporting users who wish to define and modify agent behavior in substantial ways.

Apex Example: Sequencing of Waypoints

Sequencing of waypoints by Apex has been demonstrated in simulation. This was achieved through the integration of all of the software components and messaging system (DOMS, described below) in the RIPTIDE environment. PDL was developed for Apex that solved for the path around a series of waypoints that would minimize obsolescence of the waypoints as a whole. Fig. 9 shows the moving map from the simulation with the waypoints indicated as colored symbols that turn from green, to yellow, to red as they obsolesce. In the figure, the aircraft is flying the waypoints in the sequence commanded by Apex. The sequence is continually updated as waypoints are added or removed or as new information about the waypoint obsolescence becomes available.

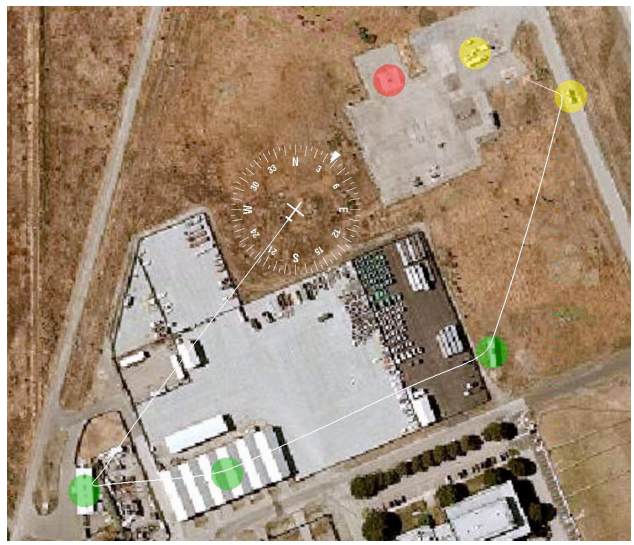

Fig. 9. Moving map showing Apex path planning (no path smoothing).

DISTRIBUTED OPEN MESSAGING SYSTEM (DOMS)

ARP requires a communication standard that can cope with the intensive data flow between the wide variety of processes necessitated by the R&S mission. Flight control, path generation, video processing, health monitoring, and mission planning all have different needs with respect to data communication bandwidth, synchronization, and quality. To meet this need, on-board and telemetry information exchange is performed using the newly-developed Distributed Open Messaging System (DOMS).

DOMS uses a publish-subscribe style message passing communications architecture. Publish-subscribe message passing is the preferred way to handle data flow from multiple asynchronous sources as is commonly found in robotics applications. The publish-subscribe technique allows data to be exchanged with little or no information about other processes in the system. This allows the modules in the system to be started or restarted in any order. It also allows the system to easily expand as more advanced hardware and software becomes available. For example, if a camera payload wants to know “where are we?”, it simply subscribes to the vehicle state message and is then notified of the current vehicle state as often as it is updated.

DOMS is designed for both local and widely distributed systems. By using message passing, DOMS makes the location of modules irrelevant. For example, modules may be running on the same CPU, a different aircraft CPU, within a separate payload, or on the ground. A communications transport daemon, domsD, is run on each computer to fa-

cilitate the exchange of information. The daemon is started before any other modules and relays messages without having to decode or convert them. If the computers are connected by one or more unreliable communication links, the transport daemon can use multiple connections or multiple transmissions to insure that the message is delivered. The transport daemon can also compress messages to improve communications efficiency. All the issues of byte order and local structure field sizes are handled transparently by DOMS, allowing a heterogeneous mix of computers. In addition, the binary messages can be efficiently logged for post processing.

Messages are published by or subscribed to using a DOMS uniform resource locator (URL), samples of which are shown in Table 2. Similar to a web browser URL, a DOMS URL contains the message type, the sending module, and the message name. Variable types handled include unsigned short integers, long integers, floats, doubles, and character arrays. Both static and dynamic arrays are handled. Optional parameters allow additional information to be specified as needed (e.g., Quality of Service (QoS) requirements). When subscribing to a message, the module and message name may utilize pattern matching to filter from all the possible messages being sent in the system.

DOMS supports structured, binary messages. The structure of a message is defined by Compact eXternal Data Representation notation (CXDR). This allows the fields of the message to be described precisely. Special notation allows strings and variable length arrays to be properly handled. Sub-structures, enumerated types, and numeric constants are also supported. A code generator creates the following code from the CXDR message description: C/C++ headers, memory allocation, packing, byte swapping, printing, and a Python language interface. The generated code accounts for the specifics of the local computer, checks for buffer overflows, supports debugging, and does not require any manual editing. To date, DOMS has been tested under the Linux, IRIX, Solaris, and Mac OS X operating systems.

A variety of communications channels may be used (TCP, UDP, Unix domain streams, etc). This allows QoS needs of specific messages to be matched with the properties of the given channel.

CONTROL LAW SOFTWARE ARCHITECTURE

The Control Law (CLAW) provides attitude stabilization and waypoint guidance control. CLAW is organized as shown in Fig. 10. There are two threads of execution. The higher priority main thread cycles at 50 Hz. The lower priority thread runs the Message Handler. Data flow in and out of CLAW by either communicating directly with the vehicle serial ports or passing DOMS messages through the Message Handler. The purpose of separating the Message Handler into its own thread was to segregate the non-deterministic network protocol portion of CLAW from the more strictly scheduled 50 Hz main thread.

Table 2. Partial listing of DOMS messages.

doms:///gpsPosition/gps/position – aircraft position values obtained from the onboard GPS receiver
doms:///gpsVelocity/gps/velocity – aircraft velocity values obtained from the onboard GPS receiver
doms:///domsMsgCharArray/basegps/corrections – GPS differential correction values obtained from the ground GPS receiver
doms:///rcDynamicState/claw/dynamicState – aircraft state values including Euler angles, position and velocity; assembled from IMU and GPS data
doms:///rcSimpleWaypoint/apex/waypoint – waypoint information sent to the waypoint follower by Apex
doms:///rcLoadWaypoint/apex/loadwaypoints – instructs the waypoint follower to start following the waypoints previously sent by Apex; also includes a bounding box which all the waypoints must reside
doms:///rcSpeed/apex/speed – sent from Apex – contains maximum speed with which the waypoint follower should fly the waypoints
doms:///rcHeading/apex/
heading – sent from Apex – contains desired heading the aircraft should assume while flying the waypoints
doms:///rcSimpleWaypoint/claw/currentWaypoint – sent to Apex– contains information from waypoint follower about waypoint currently being flown
doms:///rcSimpleWaypoint/map/selectedWaypoint – sent to Apex from the moving map – allows a human operator to instruct Apex to insert a waypoint of the operator's choosing
doms:///domsMsgChar/claw/flash – contains the type of signal used to drive the onboard flashers; onboard flashers are activated to give ground personal an indication of the operational state of the aircraft

Format: *protocol://hostname/structure_name/sending_process_name/description*

Compiler directives are used to target build types for various stages of the code development and testing. Build types differences are confined to the Input and Output modules so that each level of testing uses identical core control law code. A standalone version can be built to run independently for low level testing. A version can be built to run in the RIPTIDE environment, which is useful for higher level system testing and for checking operational procedures. Finally, the flight version can be built and run on the vehicle computer. A built-in linear test model can be enabled to perform closed loop testing for all of the build types.

DOMS messages communicated by the Message Handler include the following: 1) low-rate measurements such as GPS

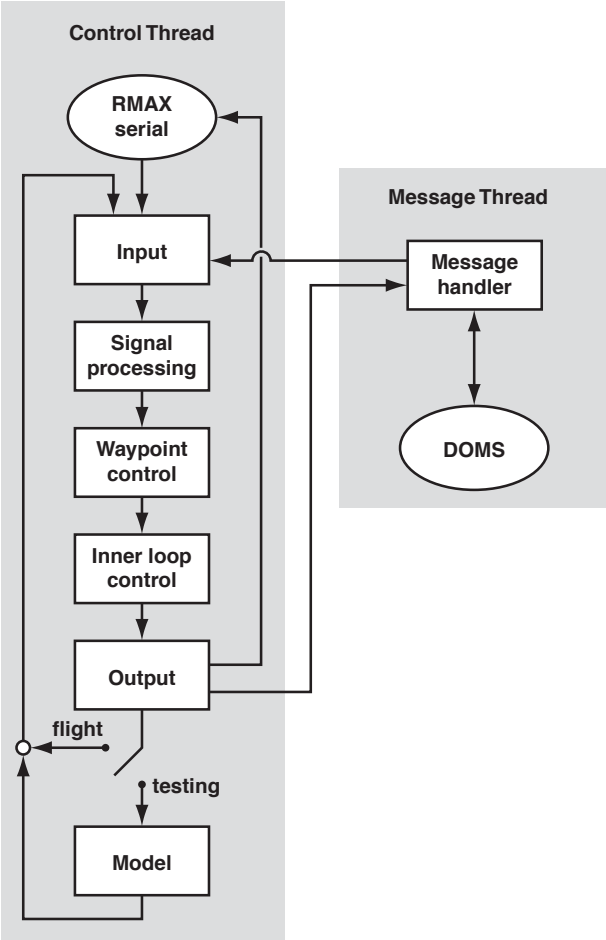

Fig. 10. Control law software architecture.

and Sonar Altitude, which are used to estimate the vehicle position and velocity; 2) operator commands to change CLAW variables; 3) telemetry data to the ground for display or recording; and, 4) guidance information from the planner in the form of waypoints.

The Input Module reads both the vehicle serial data and processes any messages arriving in the Message Handler. High-rate attitude, attitude rate, and acceleration measurements critical to the attitude control are read directly from the serial ports on each cycle of the control law. The low-rate measurements are de-queued from the buffers filled by the Message Handler and distributed to various internal data structures for use by the control law. Waypoints received by the reactive planner are also de-queued and sent to the waypoint controller. Any ground commands are also processed here.

The Signal Processing module receives the sensor input and applies the necessary filtering to generate vehicle dynamic state estimates. This module also conditions the actuator activity and other vehicle health related data.

The Waypoint Controller takes the vehicle state estimates and the waypoint data from Apex and commands the Inner Loop Controller. The inertial position and velocity are converted to attitude commands and altitude rate commands, which are sent to the Inner Loop Controller.

The Inner Loop Controller feeds back the vehicle state information to provide an ACAH control system to the Waypoint Controller. The control law is designed using single input/single output design techniques on each axis of control using stick coordinates. The stick coordinates are then transformed into actuator commands, which are sent to the Output Module.

The Output Module sends all data through either the serial port or the Message Handler. Actuator commands from the Inner Loop Control are sent though the serial port. All other low rate data are sent through the Message Handler.

As mentioned previously, CLAW has a built-in Model which can be enabled for testing purposes. The model contains the key components that significantly affect the quality of the feedback including the identified vehicle model, nonlinear kinematics, position and rate limited actuators, transport delay, sensor noise, and sensor quantization effects. The actuator commands are sent to the internal model as well as the vehicle actuators. The identified linear flight model was obtained from flight test data using established system identification methods (Ref. 13). Fig. 11 shows a frequency response comparison of the identified pitch-rate-to-longitudinal-stick linear model to flight data. Enabling the model in the flight build allows for limited hardware-in-the-loop (HIL) closed loop testing on the ground. During HIL testing the sensor measurements are overwritten with the internal model values.

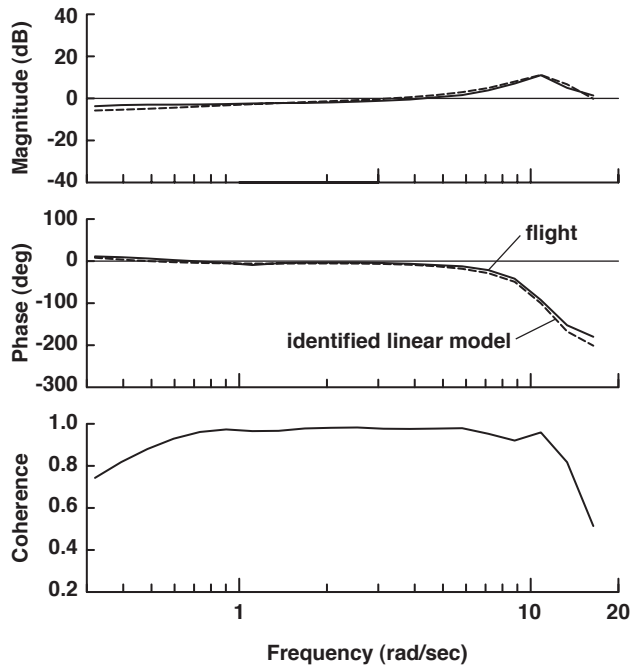

Fig. 11. Comparison of identified pitch-rate-to-longitudinal-stick linear model frequency response to the same response derived from flight data.

CLAW provides the ability to make variable changes during execution. This enables the ground operator to adjust system gains during development and is also the mechanism by which the ground operator alters the state of the control law (e.g., opening or closing a particular feedback loop). Specialized handshaking between the air and ground computers ensures that uncorrupted values within predefined bounds have been transmitted prior to being deposited. A master reset capability provides immediate reconfiguration of all altered variables to pre-flight initial states.

VEHICLE HEALTH MANAGEMENT

UAV reliability has proven to be a challenge due to their typically single-string flight control and sensor systems, and limited self awareness (Ref. 14). ARP will be addressing this area using a number of previously developed diagnostic tools. The real-time information produced by these tools will be used as input to the Apex reactive planner to aid in mission planning.

A suite of commercial tools developed by Qualtech Systems, Inc. under NASA SBIR funding provides the modeling environment as well as the real-time diagnostic routines. The Remote Diagnostic Server (RDS, Ref. 15) provides the in-flight capability, while QSI's TEAMS (Testability Engineering and Maintenance System) tool is used to develop the system model initially used for testability analysis of the system design. The model is also used by RDS for online monitoring and diagnostics. These tools have been used in other recent NASA applications for Integrated Vehicle Health Management and are included in Honeywell's open architecture definition for IVHM, developed under the Space Launch Initiative, described in reference 16. This method will serve as a baseline for other diagnostic strategies that may be applied in ARP.

A model-based approach is being used to capture system information for use in an intelligent diagnostic capability. A dependency model captures the system's configuration and observability. The multi-signal hierarchical modeling methodology uses a directed graph to capture the effects of failures in terms of their propagation paths (Ref. 17). The graph model being developed for ARP is shown in Fig. 12. The initial model tracks the hardware and software configuration and provides documentation of the available sensors and how they are utilized.

Propagation algorithms convert the graph to a single global fault dictionary for a given mode and state of the system. This dictionary contains the basic information needed to interpret test results and diagnose failures during real-time monitoring. In the real-time implementation, the sensor data must be processed to determine features that can be mapped to the health status of the component or function being monitored. Once these features are defined, tests are developed that compute the features and map them to various failure modes or off-nominal behaviors. These tests will become part of the healthPlus module which will then forward the test results to a higher-level reasoning module. The graph model is then used to analyze the propagation of the effects of off-nominal behavior to diagnose the root cause of the problem.

Using RDS to Monitor Mode Switching

One of the first goals in developing the health management system, beyond simple threshold tests, is verification that the RMAX is ready to transition between flight modes. As sensors and flight parameters are monitored during a flight, the RDS will monitor system status and advise whether the system is ready for transitioning from remote control to computer control, for example. In order to accomplish this goal, the dependency model will need to capture mode-specific behavior and be capable of discerning the correct operation of the components that are critical to the desired mode, function, and transition. During the initial development phase, the decision to transition will be made by the ground crew after manually reviewing a checklist in combination with system status checks that can be performed by CLAW. By incorporating the results of the monitoring capability that CLAW performs into the RMAX dependency model, these two check-out steps can be performed automatically and the test results analyzed by RDS. RDS will then provide a decision whether to proceed with the mode transition. Additional advanced capabilities are under investigation for incorporation into an intelligent health management module that can provide status of available functions to the mission planner.

CONCLUDING REMARKS

Autonomous helicopter operations, be they for scientific or military purposes, pose many complex challenges. The Autonomous Rotorcraft Project (ARP) has developed an autonomous helicopter research platform to identify and address those weaknesses that impact autonomous mission effectiveness the most. By demonstrating and evaluating potential solutions to these problems, ARP will provide much needed design guidance to future NASA and Army system development efforts. Key project efforts include:

1) Definition of an autonomous R&S mission that focuses development of a wide range of autonomous technologies and demonstrates the potential of future systems.

2) Integration of dual flight computers, precision GPS, analog temperature and vibration sensors, digital radio telemetry, and four digital cameras (including a wide-baseline stereo pair) into a Yamaha RMAX helicopter resulting in a system that is highly capable yet flexible enough to permit easy integration of additional sensors or technologies.

3) Integration of the Apex reactive planner providing a new framework for rapid mission definition and execution that is robust to uncertainty.

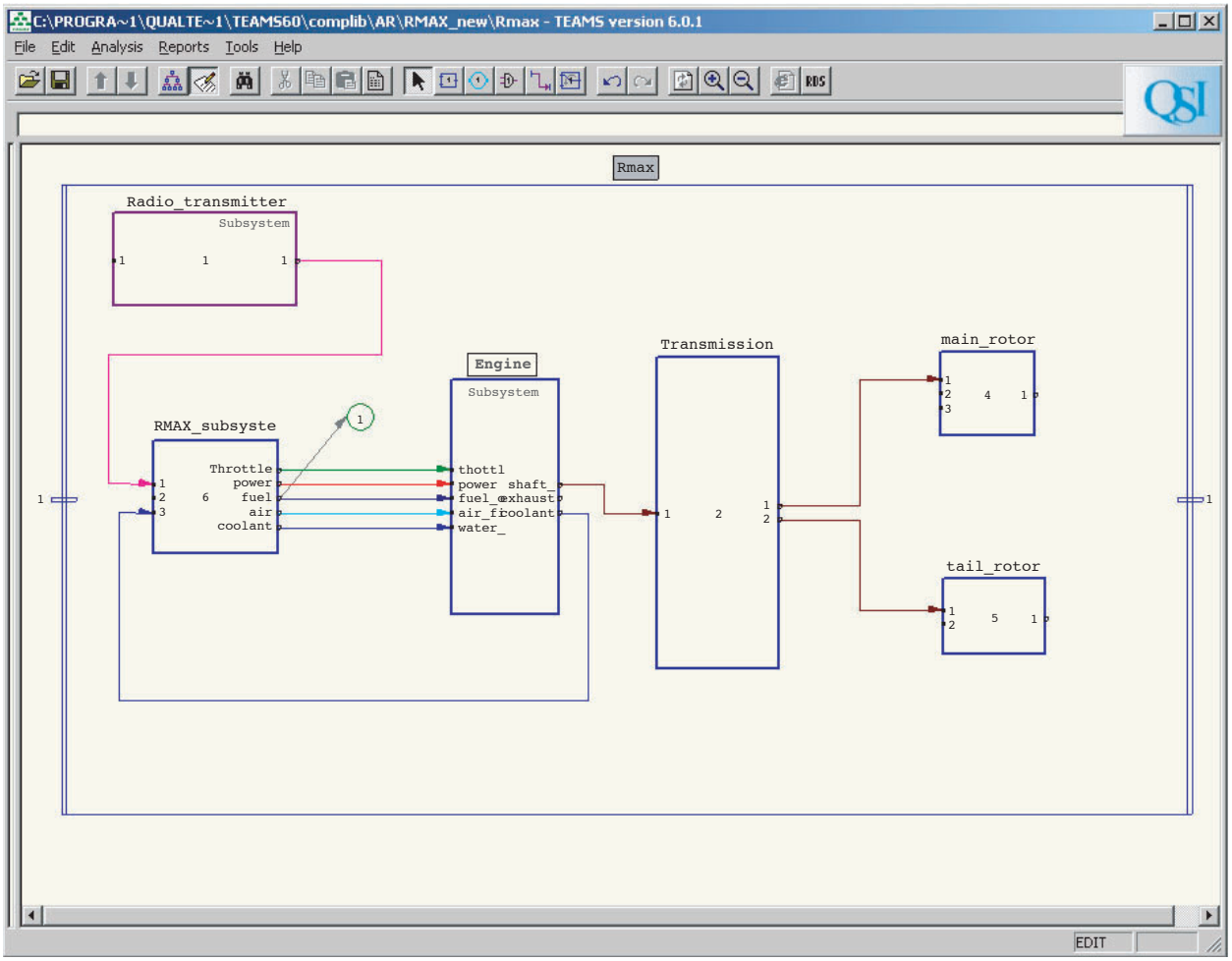

Fig. 12. Screen capture of TEAMS graph model development tool showing ARP RMAX model.

4) Development of the DOMS communication standard enabling on-board and telemetry data flow capable of coping with the wide variety of processes necessitated by the R&S mission.

5) Careful modeling of the unstable platform dynamics allowing confident and high-performance flight control.

6) Integration of the TEAMS system modeling and health monitoring software to enhance reliability and mission flexibility by providing real-time diagnostics to the mission planner.

7) Integration of the RIPTIDE simulation tool for development and operational visualization.

REFERENCES

1. Computing, Information, and Communications Technology Program Plan, www.cict.nasa.gov, October, 2001.

2. Thakoor, S., "Biomorphic Explorers," *Journal of Space Mission Architecture*, Issue 2, pp 49-79, Fall 2000.

3. Thakoor, S., "Bioinspired Engineering of Exploration Systems," NASA/DoD Second Biomorphic Explorers Workshop Bio-inspired Engineering of Exploration Systems, Pasadena, CA, December, 2000.

4. anon., "Descriptive Summaries of the Research, Development, Test And Evaluation, Army Appropriation, Budget Activities 1, 2, and 3," Department of the Army, Office of the Secretary of the Army (Financial Management and Comptroller), February 2002.

5. Balya, D., Rekeczky, Cs., and Roska, T., "A Realistic Mammalian Retinal Model Implemented on Complex Cell CNN Universal Machine," 2002 IEEE International Symposium on Circuits and Systems, Scottsdale, AZ, May 2002.

6. Voorbraak, F., Massios, N., "Decision Theory Meets AI – Qualitative and Quantitative Approaches," ECAI-98 Workshop, Brighton, England August 1998.

7. Mansur, M., Frye, M., Montegut, M., "Rapid Prototyping and Evaluation of Control Systems Designs for Manned and Unmanned Applications," American Helicopter Society 56th Annual Forum, Virginia Beach, VA, May 2000.

8. Freed, M., "Managing Multiple Tasks in Complex, Dynamic Environments," 15th National Conference on Artificial Intelligence. Madison, Wisconsin, July 1998.

9. Firby, R., “Adaptive Execution in Complex Dynamic Worlds,” Ph.D. Thesis, Yale University Computer Science Department, Technical Report 672, January 1989.

10. Gat, E., “Integrating Planning and Reacting in Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots,” 10th National Conference on Artificial Intelligence, San Jose, CA, July 1992.

11. John, B., Vera, A., Matessa, M., Freed, M., and Remington, R., “Automating CPM-GOMS,” CHI 2002 – Conference on Human Factors in Computing Systems, Minneapolis, MN, April 2002.

12. Freed, M., “Reactive Prioritization,” 2nd NASA International Workshop on Planning and Scheduling for Space, San Francisco, CA, 2000.

13. Tischler, M. and Cauffman, M., “Frequency-Response Method for Rotorcraft System Identification: Flight Applications to BO-105 Coupled Rotor/Fuselage Dynamics,” *Journal of the American Helicopter Society*, Vol 37, (3), July 1992.

14. Mount, M., “U.S. loses many military drones,” CNN News Service, January 2, 2003.

15. Deb, S. and Ghoshal, S., “Remote Diagnosis Server Architecture”, IEEE Autotest Conference, Valley Forge, PA, August 2001.

16. Dixon, R., Hill, T., Kahle, W., Patterson-Hine, A., and Hayden, S., “Demonstration of an SLI Vehicle Health Management System with In-Flight and Ground-Based Subsystem Interfaces,” 2003 IEEE Aerospace Conference, Big Sky, MT, March 2003.

17. Deb, S., et al, “Multi-Signal Flow Graphs: A novel Approach for System Testability Analysis and Fault Diagnosis,” IEEE Autotest Conference, Anaheim, CA, September 1994.