APL Spacecraft Autonomy: Then, Now, and Tomorrow

George J. Cancro

Spacecraft autonomy has a long and interesting history at APL. From humble beginnings, APL has developed and gradually increased the capability of a flexible and expressive autonomy system over three generations covering 10 years and seven spacecraft programs. Now APL is embarking on the development of a new set of autonomy systems that will meet the critical challenges of our National Security Space customers today and in the future. Development of this new set of autonomy systems will draw on lessons learned from the past, new technologies being developed today, and a four-pronged vision of what future APL autonomy systems need to achieve for National Security Space customers.

INTRODUCTION

Autonomy in a machine is the ability to act independently of human control. For unmanned spacecraft missions performed by the Space Department at APL, autonomy has grown to be defined as a specialized flight software facility designed to automatically detect and react to situations aboard the spacecraft without human intervention, usually to remedy faulted conditions or for safing (the process by which a spacecraft is placed in a safe state).

This article discusses the current state of the APL spacecraft autonomy system by examining the changes that have occurred to the autonomy facility. APL's evolution of the autonomy system over several generations provides insight and also sets the stage for our next generation of autonomy systems, which will use the historical lessons learned to move forward.

Armed with history, we can begin to look at where we are headed at present and where we should head in the future. This article outlines the current direction of autonomy systems at APL and discusses the future direction by examining all spacecraft onboard functions that potentially could be automated. From this list of functions, four key themes are extracted and described in terms of benefit and effect on future National Security Space (NSS) missions.

AUTONOMY: THE PAST AT APL FROM ACE TO STEREO

The story of autonomy in APL spacecraft occurs over three generations, beginning with the Advanced

Composition Explorer (ACE) mission, during which autonomy was first separated from hard-coded software, and ending with the Solar TErrestrial RElations Observatory (STEREO), which is the most recent mission launched by APL. These generations cover 10 years and seven spacecraft and are described in detail below.

Generation 1 (ACE)

The ACE spacecraft launched in August 1997 with the goal of understanding and monitoring solar activity.[1] The ACE autonomy system, in conjunction with hardware-based fault detection and reaction and together with the command and data handling (C&DH) and power subsystems, formed the overall ACE safing strategy. This autonomy system was responsible for preparing the spacecraft for first contact, monitoring component health, monitoring overall spacecraft attitude and maneuver health, and maintaining proper spacecraft component on/off configurations and other autonomous actions to support the recorder and hardware-based reactions.[2]

The ACE autonomy system, which was a facility of C&DH software, was based on a set of autonomy rules. These rules take the form of "if-then" statements that can be loaded into fixed-size memory locations known as bins. When the autonomy system is running, it scans the rules at a regular interval, evaluating each rule in turn and executing any that evaluate to "true."[3]

The "if-section" of an autonomy rule is formulated as one of six conditional types (equal to *A*, not equal to *A*, greater than *A*, less than *A*, within a range of *A* to *B*, outside a range of *A* to *B*), and the "then-section" consists of a spacecraft command to issue if the conditional is true for a predefined number of evaluations. To program an autonomous behavior, the autonomy designer would construct a rule by defining the telemetry point (a section of the spacecraft's telemetry data block representing a spacecraft sensor value), defining a mask of the telemetry point if needed, selecting the conditional type, defining the *A* and *B* values for the conditional types, defining the number of true evaluations before a command is executed, and selecting the command to issue.

The command selected to issue could be a single command or a call to a block of commands to be run in sequence. The sequence of commands could also include pauses in the sequence to provide relative timing of commands. All commands issued from the autonomy facility, whether single commands or the command sequence from a block, are executed at the same priority. Therefore, only a single autonomy rule could control the spacecraft at one time.

Development of the ACE autonomy system established the separation between rules and hard-coded autonomy at APL. Before this development, autonomous behavior was nonexistent or was directly written into the C&DH software for the spacecraft. This rule-based approach to meeting autonomy requirements allowed C&DH design to proceed, even when autonomy conditions and actions had not been fully specified at the mission level.

Generation 2 (NEAR/TIMED/CONTOUR)

The next generation of APL spacecraft autonomy systems modified the ACE autonomy design by increasing the functionality and expressiveness of the autonomy in response to the increased mission complexity.

First, the expressiveness of the conditional portion of the rule was expanded. Autonomy rules for the Near Earth Asteroid Rendezvous (NEAR) Shoemaker spacecraft, launched in 1997 to study and eventually land on the asteroid Eros,[4] were the logical AND or OR combination of two ACE rule expressions, thereby doubling the capability of ACE. Comet Nucleus Tour (CONTOUR), launched in 2002 to understand and assess the diversity of two comets, and Thermosphere, Ionosphere, Mesosphere Energetics and Dynamics (TIMED), launched in 2001 to explore the Earth's mesosphere and lower thermosphere,[5] quadrupled the capability by enabling logical combinations of four ACE expressions in each rule. In addition to expressiveness, readability was enhanced by adding another facility (called arithmetic checks) that performed conversions of telemetry point values into engineering units. For example, instead of specifying rules as "IF telemetry_point_5 > 3124 . . . ," rules could be specified as "IF imu_power > 14 W. . . ."

Even with this expansion of capability, the number of rules on each successive mission continued to grow. ACE had 64 rules, NEAR had 165 rules, TIMED had 256 rules, and CONTOUR had 259 rules. The number of autonomy system responsibilities was growing, and the complexity of the responses was increasing. To handle the growth in complexity of responses, conditional execution features were added to the APL autonomy system by allowing autonomy rules to enable or disable other autonomy rules. This allowed one autonomy rule to detect a fault and then enable a set of rules to deal with the fault depending on the current state of the system.

The increase in autonomy system rules was also a result of APL's autonomy facility having taken on more than fault management and safing. For example, TIMED used the autonomy rule facility to automate routine operations.[6] To support the increased range of responsibilities in terms of criticality, multiple levels of priority were added to the command execution of autonomy responses in this generation of autonomy systems. In this manner, the response to a higher-priority fault could preempt a lower-priority fault response or automated operations action currently being executed.

The ability to modify rules was also extended in generation 2. Instead of being able to modify autonomy rules only before launch, generation 2 system rules also could be modified by command after launch. This approach enabled APL operators to modify rule definitions at any time in the program, granting missions the flexibility to handle postlaunch anomalies and changes in operations.

Even though the addition of conditional execution and priority responses solved problems faced by autonomy developers within generation 2, we now believe this was the beginning of the end of the rule-based system. A good example of the reasons for moving away from the rule-based approach was evident in the NEAR mission: "What seemed at first to be a simple rule-based design actually became quite complex when it came to defining the checks and command responses needed to coordinate safing for all spacecraft subsystems."[7] Coordinating multiple rules to implement system-level functions also drove the testing time necessary to verify the rule implementations.

Generation 3 (MESSENGER/New Horizons/STEREO)

The next generation of APL spacecraft autonomy systems responded to the autonomy designers' resistance to the restrictiveness of the conditional portion of the autonomy rule by expanding expressiveness again. The six conditional types used on ACE were replaced by a generic reverse Polish notation (RPN) expression. This enabled designers to place in rule expressions any combination of arithmetic and Boolean operators and any number of telemetry point operands. Arithmetic checks, used in generation 2 to increase readability by translating raw telemetry to engineering units, were replaced by a new facility called computed telemetry, which enabled designers to use RPN expressions to convert telemetry or perform calculations. In August 2003, M. Gomez presented a complete description of the generation 3 autonomy system, "A Typical Spacecraft Autonomy System."[8]

At this point in the evolution, the number of rules in a given system began to decrease. CONTOUR, the last generation 2 system, had 259 rules, but the generation 3 systems, Mercury Surface, Space Environment, Geochemistry, and Ranging (MESSENGER), which launched in 2004 to conduct the first orbital study of Mercury;[9] New Horizons, which launched in 2006 to be the first spacecraft to study the Pluto–Charon system;[10] and STEREO, which launched in 2006 to capture and study the Sun in three dimensions,[11] had 208, 126, and 156 rules, respectively. However, hidden in this decrease were more increases in complexity, because the RPN system allowed more operands in expressions. For example, CONTOUR averaged 2.4 operands per rule, whereas MESSENGER averaged 10.2 operands per rule. Multiplying the number of rules by the number of operands demonstrates that MESSENGER was approximately 3.5 times more complex than CONTOUR.[12] This hidden complexity continued the trend of reduced system-level design understandability and increased the test time necessary to ensure that system-level safety was maintained.

Trends Across Three Generations

Taking a step back and examining the trends over multiple years and missions reveals three major trends.

First, what started off as a simple system incrementally grew to a fully featured system with great amounts of flexibility and expressiveness. Each feature added increased ability to meet mission complexity; however, the drive to more and more complexity has pushed the autonomy rule concept to its practical limits, exposing the tradeoff between simplicity at the individual rule level and complexity at the system level. In the end, autonomy designers' desire for more autonomy features and expressiveness resulted in unforeseen consequences on overall system complexity and impacted the time necessary for testing.

Second, the autonomy responsibilities for fault management and safing defined on ACE remained in all generations of spacecraft. For example, each generation developed autonomy rules to handle first contact and component health monitoring and to maintain system-level configurations of component on/off states. Subsequent generations increased the extent of these responsibilities and also added new responsibilities in terms of fault protection, but these core responsibilities remained. What makes this interesting is that, despite the similarity of functionality, no reuse in the rules themselves occurred. The implementations of the same responsibilities did not carry over from one generation to the next or even from one mission to another within a generation.

Third, beginning with a single rule responsible for recorder management on ACE and extending into automating routine operations on TIMED and handling instruments and operational modes in generation 3 systems, the autonomy rule facility has taken on an increasingly important role outside of the initial intent of a fault management facility. What started out as an extremely limited set of responsibilities on ACE became a large set of responsibilities by generation 3. Over the years, the flexibility of the rule-based system became more and more enticing to noncritical faults, instrument management, and then to automating operations.

AUTONOMY: THE PRESENT AT APL

Despite the problems with these trends, there is no going back. We cannot return to the ACE system for future missions because the expectations of the level of autonomy on missions have increased and the complexity of missions continues to increase as well. Instead, we must now turn to combating the unintended consequences of the desired flexibility and expressiveness: lack of reviewability, lack of reuse, and difficulty in testing.

ExecSpec

Over the last 3 years, APL has invested independent research and development funds in the development of the next generation of onboard autonomy systems. This development was motivated by the need to remedy the unintended consequences described previously without losing the flexibility to modify autonomy at any time in the mission and without losing the expressiveness required for complex space missions. The result of this development is a system called ExecSpec (short for Executable Specification). ExecSpec is a new visual programming approach to development of autonomy systems that enables a system designer to visually create and execute high-level spacecraft functionality and autonomous behavior in the form of uploadable diagrams (Fig. 1).[13]

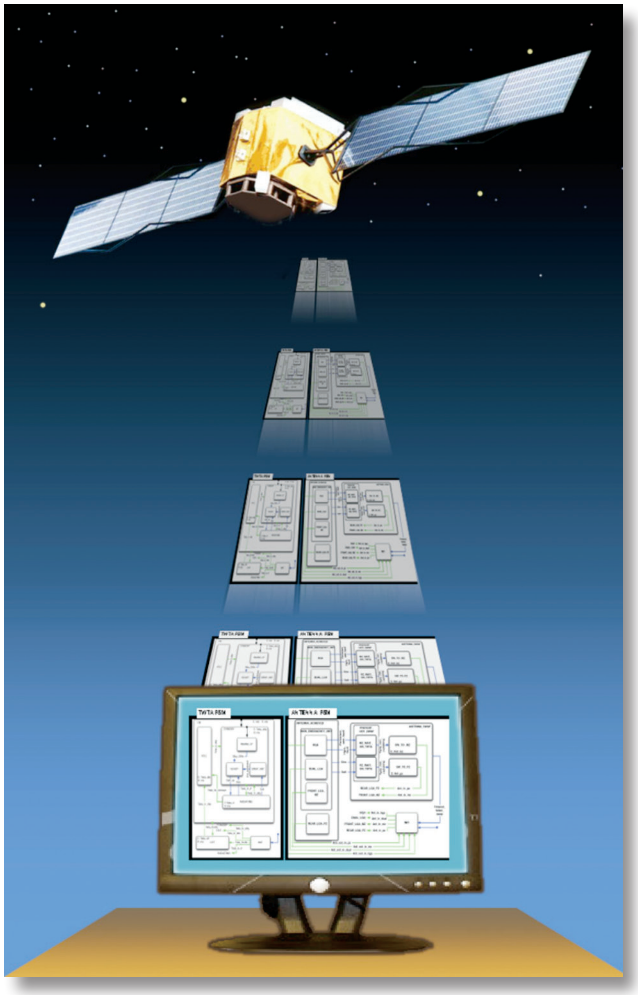


Figure 1. ExecSpec diagrams showing that desired functionality can be uploaded directly to the ExecSpec flight component within the spacecraft for execution.

Comprehensible Context Through the Entire Life Cycle

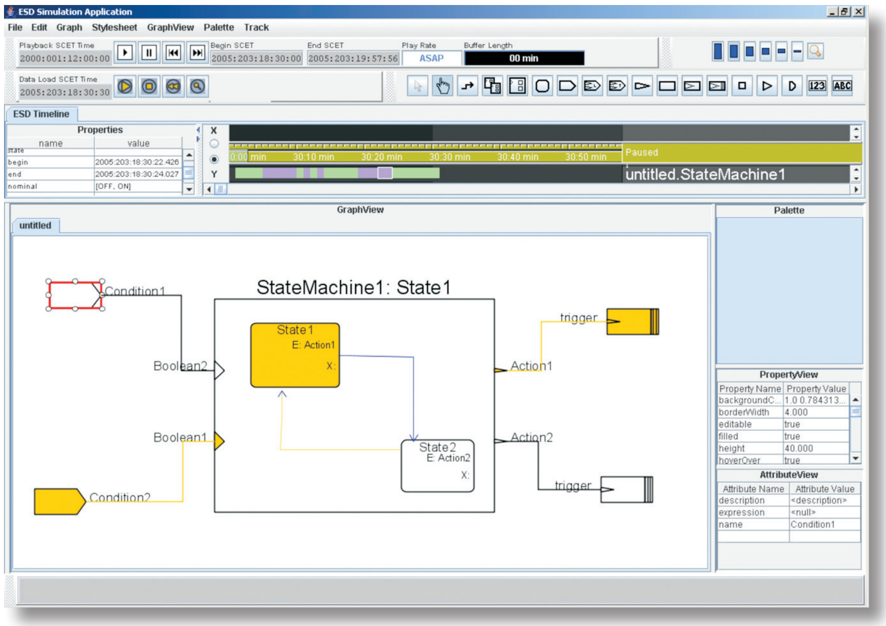
ExecSpec diagrams are based on finite state machines and make it easy for non-software experts such as system engineers, domain experts, and operators to understand the onboard functionality directly, improving the design quality and the efficiency of the design process.[14] In addition, this easy-to-understand context is maintained across the entire program life cycle. The diagrams that are used to define the design and review the implementation are the same diagrams that are used to operate the spacecraft and monitor the autonomy system telemetry. For example, during operations the diagrams are animated based on spacecraft telemetry such that operators can visually monitor the autonomy behavior during operations.

Advanced Simulation/Test Capability

ExecSpec contains two forms of advanced testing to provide mechanisms to test highly complex autonomy systems. The first is an advanced simulation capability that enables interactive testing and debugging, as shown in Fig. 2, with which an operator can test the design by interacting with it through modifying system inputs and monitoring system outputs visually. This enables a rapid design-and-test cycle that improves the design reliability and shortens the time required to produce an autonomy system.

The second testing capability is automated verification granted by combining ExecSpec with NuSMV, an automated model-checking tool.[15] This capability, shown in Fig. 3, compares the design to the project requirements by performing an exhaustive search to find counterexamples in which the design violates requirements. The benefit of this feature is the ability to rapidly test autonomy requirements. Our initial research into this effort[16] demonstrated examples of requirements from the NASA STEREO mission being tested at a rate of up to one requirement per second on a model of the STEREO autonomy system developed in ExecSpec. In comparison, the current rate for humans performing acceptance testing of autonomy systems on the NASA STEREO mission was 66 requirements over 12 months, using 6 staff months of effort, or 1 requirement per 14 staff hours.

Although at face value the benefit is large, model checking cannot be seen as a silver bullet because the technique becomes intractable with large models, it is limited by the contents of the model in comparison to the actual system, and it is not a substitute for testing on actual spacecraft hardware. However, model checking does provide an additional testing resource that was not at our disposal in the past, allowing us to combat the problem of complexity and adverse interactions within autonomy systems.



Playback SCET Time: 2000:001:12:00:00
Begin SCET: 2005:203:18:30:00
End SCET: 2005:203:19:57:56
Data Load SCET Time: 2005:203:18:30:30

Properties

name	value
state	
begin	2005:203:18:30:22.426
end	2005:203:18:30:24.027
nominal	[OFF, ON]

0:00 min 30:10 min 30:20 min 30:30 min 30:40 min 30:50 min

PropertyView

Property Name	Property Value
backgroundC...	1.0 0.784313...
borderWidth	4.000
editable	true
filled	true
height	40.000
hoverOver	true

AttributeView

Attribute Name	Attribute Value
description	<description>
expression	<null>
name	Condition1

Figure 2. Screenshot of ExecSpec demonstrating how state machine systems can be tested directly in the visual tool, displaying the current state in the diagram view and state history in the timeline view.

Reuse Through Prototype Instantiation

One of the most powerful techniques of software engineering is the use of reusable software components that can be assembled in various ways to form larger systems. In ExecSpec, this can be accomplished through a prototype-instance methodology by which a set of prototypical components can be developed, stored in a library, and then copied and interconnected to form an overall system, as shown in Fig. 4. This feature enables reuse in APL autonomy systems and dramatically decreases the time required to develop systems.

ExecSpec Benefits to NSS

Operationally Responsive Space

The concept of Operationally Responsive Space (ORS) proposes the fielding of spacecraft assets, from concept to launch, in weeks. By using the ExecSpec system, a new ORS spacecraft autonomy system can be rapidly assembled from a diagram library and tested by using the visual and automated techniques described previously in this article. After all testing is completed, the design is loaded directly into the ExecSpec flight component, which is a generic diagram interpreter that does not change from mission to mission. The result is an autonomy system that can realistically meet ORS development timelines.

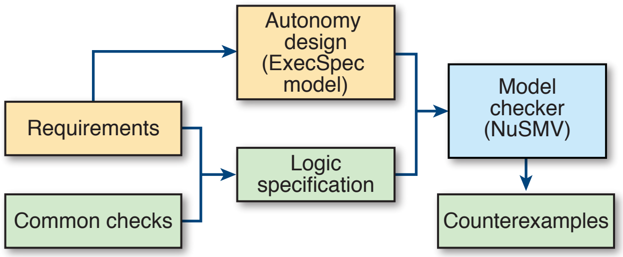


Figure 3. Model-checking process for ExecSpec diagrams using NuSMV, an automated model-checking tool.

Increasing the Survivability and Usability of Space Assets

Space assets imbued with the ExecSpec system will be flexible to a changing environment and a changing mission. Currently the response to component failures after launch or mission changes is implemented by operational workarounds, which drive up the cost and complexity of the operations and limit onboard functionality. By using the ExecSpec features, changes to spacecraft functionality can be developed, fully tested, and uploaded to a vehicle after launch. The end result is new tactical capability, resulting in an increase in spacecraft survivability and an increase in the usable duration of space assets.

AUTONOMY: THE FUTURE AT APL

To meet the critical challenges that our nation will face in the future, we must look beyond our historical and present developments to new technologies and concepts that will meet future needs of sponsors. To do this, we have performed a taxonomy analysis of all functions that can be automated aboard spacecraft. Of the total set of functions, we selected a subset that we believe is of interest to NSS customers. The selected functions were then grouped into four themes, whereby each theme possesses three increasingly complex functional steps that eventually lead to the desired end capability. These four themes are as follows:

1. Fault detection and recovery,
2. Spacecraft as extension of the non-expert user,
3. Streamline operations and enable multiples, and
4. Target of opportunity.

The themes and functional steps are displayed in Fig. 5 and described in detail in the subsections below.

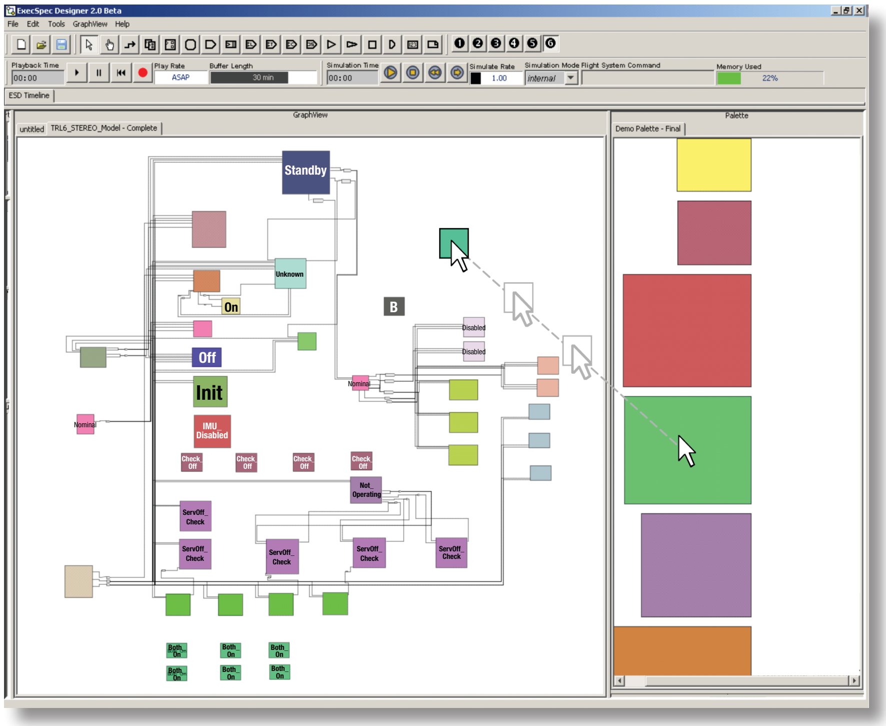


Figure 4. Screenshot of ExecSpec demonstrating instantiation of prototype components through drag-and-drop functionality.

Theme 1: Fault Detection and Recovery

Fault detection and recovery is the original application for onboard autonomy because faults can occur at any time and spacecraft are not always in contact with ground operators. Historically, fault detection and recovery resulted in driving the system to a safe state. In the future, fault detection and recovery must move in the direction of recovering the spacecraft from a fault into an operational state. In essence, the spacecraft must autonomously reconstruct an operational system from a faulted one. This will enable spacecraft to continue their missions and maintain high levels of availability to users on the ground.

Finally, as the number of threats to on-orbit spacecraft increases, faults may be induced by hostile actors outside the spacecraft. In this case, autonomy must be operationally responsive to these threats (i.e., self-preservation through autonomous reconfiguration) such that external threats can be detected, communicated to ground operators or other spacecraft, and handled by the spacecraft modifying itself or its operational environment to be able to continue the mission. Therefore, in the near future, fault detection and recovery should be considered part of the overall space situational awareness and defensive counterspace function.

Theme 2: Spacecraft as Extension of the Non-Expert User

In addition to autonomous fault detection and recovery, current spacecraft also act autonomously outside of ground contact to execute time-based scientific or engineering operations. Historically, time-based operations have been executed by spacecraft operational staff with primitive scripts or time-tagged commands, usually in 2-week scheduling periods. In this architecture, the operations staff becomes the gatekeeper of spacecraft activity whereby users can submit requests that eventually are translated into spacecraft time-based commands. Ideally, in the future, onboard autonomy should enable spacecraft operation to be driven tactically by non-expert users. The first step toward this goal would be an agile and flexible tasking system that would enable adaptive planning cycles on the order of a day or an orbit. This would replace the scripted data acquisition cycles with a system that is directly responsive to an operational theater commander. The final step toward the spacecraft becoming an extension of the non-expert user is the ability to autonomously request and view data in context. For example, a field commander requiring surveillance of areas of future operation should be able to circle an area of a map to ask for updated satellite imagery of that area. The resulting surveillance from the satellite should appear to the user as updated images in the area that the commander identified. In this manner, the user can request and view data in the context (the map) in which the user normally works.

Theme 3: Streamline Operations and Enable Multiples

All spacecraft perform a set of one-time and routine maintenance operations on orbit. These activities include on-orbit check-out, contact scheduling, calibration, and long-term assessments. Historically, these operations have been performed manually by operations staff. APL has automated some of these routine operations to reduce overall operational costs. Future autonomy systems should continue to streamline operations to reduce cost and increase speed and should strive to enable the operation of multiple spacecraft with small operational teams. To achieve these goals, autonomy development should focus on the ability to rapidly and

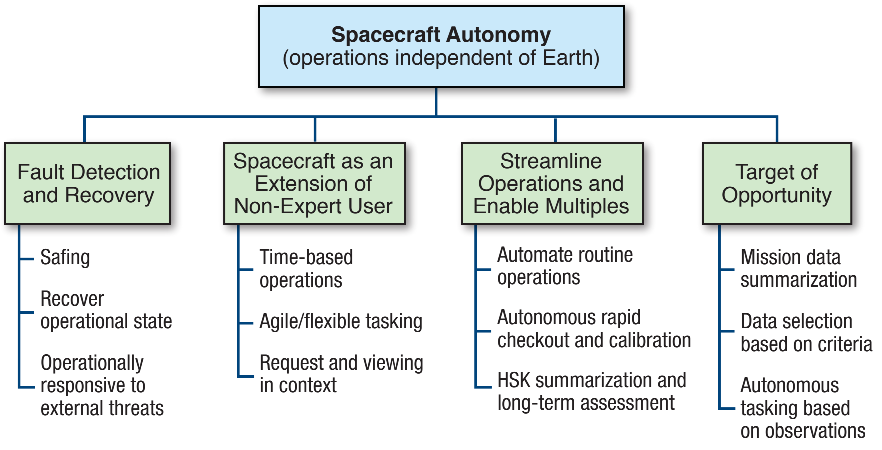

Spacecraft Autonomy
(operations independent of Earth)
Fault Detection and Recovery
Safing
Recover operational state
Operationally responsive to external threats
Spacecraft as an Extension of Non-Expert User
Time-based operations
Agile/flexible tasking
Request and viewing in context
Streamline Operations and Enable Multiples
Automate routine operations
Autonomous rapid checkout and calibration
HSK summarization and long-term assessment
Target of Opportunity
Mission data summarization
Data selection based on criteria
Autonomous tasking based on observations

Figure 5. Autonomy taxonomy for NSS. HSK, housekeeping.

autonomously check out spacecraft to speed up the time from launch to operational readiness. Current check-out times are on the order of several weeks to a month. For spacecraft to be truly operationally responsive, greater speed must be achieved from development all the way to readiness. Because readiness includes calibration of instruments, one-time and periodic calibrations should be automated. Finally, all issues with routine operations become more complex for multiple spacecraft constellations. Autonomy systems should be able to reduce the burden on operational teams. Housekeeping data summarization and long-term health assessment is one area that could provide savings. For example, if the spacecraft could autonomously alert operators about interesting artifacts in housekeeping, the constellation bandwidth required for operations would decrease, as would the workloads of the operators.

Theme 4: Target of Opportunity

As noted previously, target designation and acquisition historically has been accomplished in 2-week schedules developed on the ground and then executed using multiple time-based commands on board. All the resulting data are then downlinked to ground users at the next contact opportunity. The amount of data collected is therefore limited by the downlink bandwidth. In the future, the capacity of sensors to produce data will rapidly overcome the bandwidth available to return the data, forcing operators to be selective about what they acquire and return. To address this challenge, autonomy can be used to prioritize onboard data in a wide range of options, from providing sensor data summaries so that ground operators can select relevant data to autonomously selecting data for downlinking on the basis of predefined criteria. The ultimate extension of this concept would be the ability for the spacecraft to autonomously acquire data on the basis of opportunity or prior observation. In such a scenario, a spacecraft could take data in a discovery mode and then autonomously switch from discovery mode to an active high-rate mode to capture relevant data predefined by mission operators. In effect, the spacecraft could then acquire data desired by ground operators or users without the user specifically requesting the exact information.

APL SPACECRAFT AUTONOMY ROAD MAP

Armed with the four themes described in the preceding sections, APL is developing autonomy capabilities to achieve the goals and functional steps outlined above. With lessons learned from the past, we have concluded that the predisposition to use the existing autonomy development facility to implement all of the desired autonomous functionality has expanded it to the point of overcomplexity. Therefore, our plan is to implement desired autonomy functionality as multiple, separate, specialized applications rather than follow a one-size-fits-all approach. This approach will better handle the growing complexity of desired functionality without overcomplicating existing capabilities and will also provide a mechanism for incremental improvement.

Currently, the ExecSpec system is envisioned to meet theme 1 (fault detection and recovery). The next APL mission will use this system as the basis for fault protection autonomy. The abilities of the system to provide safing, recover the operational state, and reconfigure space systems on the basis of external threats will benefit all space missions and provide the technology to complement onboard space situational awareness detection sensors and algorithms. In addition, the ability to rapidly construct autonomous systems through drag-and-drop reuse will increase the speed of autonomy development to the level necessary to support ORS.

The next target for APL will be in the development of an agile and flexible tasking system. In FY2009, APL began research to develop an agile tasking system that is based on the use of hierarchical simple temporal networks. We believe that the ability to move satellite tasking from strategic to tactical users is key to the concept of ORS and is also useful to other imagery-intensive organizations such as the National Reconnaissance Office.

Finally, APL is also experimenting with real-time feature extraction and data-mining techniques to begin investigating the aspect of theme 4 by which data selection would be based on criteria. Coupled with the ExecSpec system and the agile tasking system, we believe that a unique and powerful autonomous platform can be created. This future platform would be able to analyze data taken for desired criteria, the agile task itself, and then reconfigure itself to continue the mission.

REFERENCES

[1]*ACE Mission Factsheet*, http://sd-www.jhuapl.edu/ACE/ACE_FactSheet.html (accessed 4 Apr 2010).

[2]Chiu, M. C., et al., "ACE Spacecraft," *Space Sci. Rev.* **86**(1–4), 257–284 (1998).

[3]Bogdanski, J. F, Conde, R. F., and Williams, S. P., *ACE Spacecraft Command & Data Handling Component Specification*, JHU/APL Document 7345-9030 (8 Mar 1996).

[4]NEAR Mission website, http://near.jhuapl.edu/ (accessed 4 Apr 2010).

[5]TIMED Mission website, http://www.timed.jhuapl.edu/WWW/index.php (accessed 4 Apr 2010).

[6]Harvey, R. J., "TIMED Autonomy System," *Johns Hopkins APL Tech. Dig.* **24**(2), 201–208 (2003).

[7]Stott, D. D., et al., "The NEAR Command and Data Handling System," *Johns Hopkins APL Tech. Dig.* **19**(2), 220–234 (1998).

[8]Gomez, M., "A Typical Spacecraft Autonomy System," International Conference on Machine Learning (ICML) Workshop on Machine Learning Technologies for Autonomous Space Applications, Washington, DC (21–24 Aug 2003).

[9]MESSENGER Mission website, http://messenger.jhuapl.edu/ (accessed 4 Apr 2010).

[10]New Horizons Mission website, http://pluto.jhuapl.edu/ (accessed 4 Apr 2010).

[11]STEREO Mission website, http://stereo.jhuapl.edu/ (accessed 4 Apr 2010).

[12]Hill, A., *Autonomy Metrics from the TIMED, CONTOUR, MESSENGER, New Horizons and STEREO Missions*, Technical Memorandum SIE-07-040, JHU/APL, Laurel, MD (Aug 2007).

[13]Cancro, G., Innanen, W., Turner, R., Monaco, C., and Trela, M. "Uploadable Executable Specification Concept for Spacecraft Autonomy Systems," in *Proc. IEEE Aerospace Conf.*, Big Sky, MT, pp. 1–12 (2007).

[14]Turner, R., Hooda, S., Gersh, J., and Cancro, G., "ExecSpec: Visually Designing and Operating a Finite State Machine-based Spacecraft Autonomy System," 9th International Symposium on Artificial Intelligence, Robotics and Automation for Space, Pasadena, CA (26–29 Feb 2008)

[15]NuSMV website, http://nusmv.fbk.eu/ (accessed 4 Apr 2010).

[16]Pekala, M., and Cancro, G., "Verifying Executable Specifications of Spacecraft Autonomy," 9th International Symposium on Artificial Intelligence, Robotics and Automation for Space, Pasadena, CA (26–29 Feb 2008)

The Author

George J. Cancro is the Assistant Group Supervisor of the Embedded Applications Group in the Space Department. He holds a B.S. in engineering science from Penn State University and an M.S. in mechanical engineering–astronautics from the George Washington University. Before joining APL in 2002, he worked at the NASA Jet Propulsion Laboratory and NASA Langley Research Center on projects such as Mars Global Surveyor Aerobraking and the Dawn Mission to Vesta and Ceres. Since joining APL, he has worked as a systems engineer on the MESSENGER, New Horizons, and STEREO missions; a project manager for the NASA SmallSat project; and a principal investigator of two research projects in the areas of autonomy and telemetry visualization. He is currently a principal investigator of a research project investigating spacecraft tactical commanding and an advisor to the NASA Constellation Program in the area of fault detection, isolation, and recovery. His areas of interest include modular software, hardware/software architectures, fault protection, and spacecraft autonomy. His e-mail address is george.cancro@jhuapl.edu.

George J. Cancro

The *Johns Hopkins APL Technical Digest* can be accessed electronically at **www.jhuapl.edu/techdigest**.