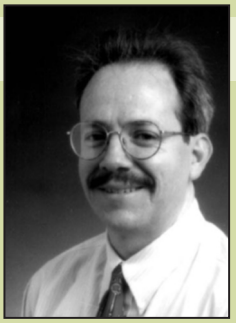


Editor: Richard Doyle
Jet Propulsion Lab
rdoyle@jpl.nasa.gov

Space Shuttle Ground Processing with Monitoring Agents

Glenn S. Semmel, Steven R. Davis, Kurt W. Leucht, Daniel A. Rowe, and Kevin E. Smith, *NASA Kennedy Space Center*
Ladislau Bölöni, *University of Central Florida*

"3…2…1…and liftoff of Space Shuttle Discovery!" Those few words signaled the shuttle's return to flight after two and a half years' hiatus. For engineers and technicians at NASA's Kennedy Space Center in Florida, those words also represented the culmination of hundreds of thousands of ground processing person-hours that precede every shuttle launch.

The Space Shuttle system is among the most complex machinery ever built and maintained by humans. Just prior to launch, 4.5 million pounds of combined weight—orbiter, external tank, solid rocket boosters, and propellants—sit on the launch pad (see figure 1). Eight and a half minutes later, the orbiter is cruising at 17,440 mph, 59 nautical miles above Earth and still hurtling on its upward climb to orbit.

Editor's Perspective

As with aircraft, launchings and landings are arguably the most exciting and certainly the most intense events for spacecraft missions. As operators of the US's spaceport, the personnel at Kennedy Space Center are the experts at launching spacecraft, whether to Earth orbit or to a remote destination in the solar system. Understandably, the role of automation and decision support for such critical activities has been approached conservatively as a rule—decision support is primarily applied to non-real-time analysis functions and automation is usually reserved only for executing well-understood contingencies when the limits of human reaction time is a factor. Nonetheless, new approaches to automation and decision support—including agents technology—are making their way into launch services functions at KSC, due to the efforts of an enterprising team. This issue's article discusses their early successes and future prospects. *—Richard Doyle*

To ensure that all shuttle systems are ready for launch, KSC engineers monitor tens of thousands of telemetry measurements. This intense data monitoring occurs constantly during ground processing, peaking in the weeks leading up to launch. KSC engineers developed and deployed a software agent—the NASA Engineering Shuttle Telemetry Agent—to assist in this around-the-clock monitoring. NESTA operates 24 hours a day, 7 days a week, sniffing out predefined data patterns of interest and instantly notifying shuttle engineers via email or wireless page.

Prelaunch ground checkout

NASA KSC is responsible for the Space Shuttle's prelaunch ground checkout. KSC's Launch Processing System provides facilities for shuttle system engineers, contractors, and test conductors to command, control, and monitor space vehicle systems from the start of shuttle interface testing through various phases including terminal countdown and launch.

LPS continuously monitors the shuttle and its ground-support equipment such as the hardware that loads propellants. Subsystems with vehicle responsibilities communicate information directly to and from the shuttle computer systems. Subsystems with ground-support equipment responsibilities communicate through hardware interface modules that connect to the numerous ground-support systems. Each module can interface to approximately 240 sensors or controls. Overall, some 50,000 temperatures, pressures, flow rates, liquid levels, turbine speeds, voltages, currents, valve positions, switch positions, and many other parameters must be monitored and controlled. Figures 2 and 3 provide glimpses of the ground communication infrastructure necessary for this extensive instrumentation.

Using LPS, NASA engineers and contractors at KSC certify that the Space Shuttle's ground checkout meets program specifications. For over 25 years, engineers have used LPS to verify Space Shuttle flight readiness and con-

1541-1672/06/$20.00 © 2006 IEEE
Published by the IEEE Computer Society

trol launch countdown. It has performed superbly, and recent upgrades ensure its continuance for many more years. However, the system architecture was not changed, so the system and display software remains basically the same. As a result, the level of situational awareness has not kept pace with more modern software technologies.

Day-to-day shuttle operations, checkout, and maintenance are contracted out, so contractors are LPS's primary users. NASA shuttle engineers are civil service employees who oversee the contractors. Given LPS limitations and resource scarcity, those engineers needed a tool to provide more insight to and situational awareness of the contractors' work. They needed a tool to complement LPS that could autonomously and continuously monitor shuttle telemetry data and automatically alert them when predefined criteria had been met. In the latter half of 2003, the NESTA software tool emerged to improve insight into shuttle ground processing and increase situational awareness.

Figure 1. Space Shuttle Discovery leaving the launch pad for STS-114's "Return to Flight" mission.

NESTA objectives

Figure 4 shows how LPS-processed measurements are distributed on a LAN as the Shuttle Data Stream,[1] providing real-time telemetry from shuttle ground-processing and launch operations. Various monitor-only applications that require shuttle engineers' full attention use this data stream to observe events and detect anomalies. NESTA's primary objective is to continuously and autonomously monitor this telemetry stream, automatically alerting NASA engineers in near real time when the data meet predefined criteria. Types of monitoring criteria include expected operational events or milestones (such as vehicle power-up) as well as unexpected events or failures (such as large differences between redundant sensor values).

NESTA acts as a software agent for the NASA engineer. For this discussion, we define an agent as rule-based, autonomous software that reacts to its environment and communicates results to a human—the NASA engineer. NESTA's primary objectives include the following:

- Apply engineer-specified rules to measurements published in the telemetry stream.
- When engineers' rules are satisfied, generate near-real-time notifications and alerts in the form of emails or wireless pages. Notifications might include a text message and measurement values, and they might go out to multiple users.
- Monitor up to four separate telemetry sources simultaneously, including four control rooms used for checkout and launch of the shuttle and its components.
- Process multiple types and subtypes of measurements and read-only commands including discretes (Boolean measurements), analogs (floating-point measurements), and digital patterns (integer measurements).

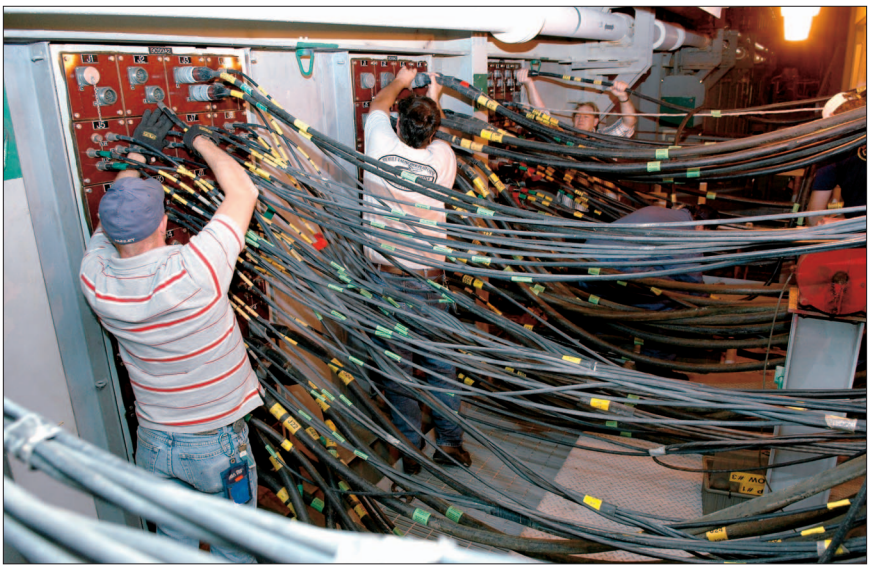


Figure 2. During a rollback, workers demate the fittings between the pad and the mobile launcher platform, allowing the MLP to return with Space Shuttle Discovery to the vehicle assembly building to have its external tank replaced. The fittings provide ground electrical power and connections for vehicle data and communications.

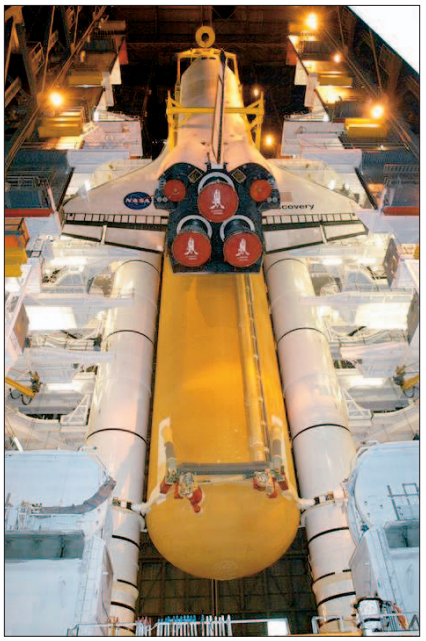


Figure 3. Inside the vehicle assembly building, NASA technicians lower the orbiter Discovery in front of the solid rocket booster and external tank already stacked on the top of the mobile launcher platform. After Discovery has been mated to the external tank and solid rocket booster assembly on the MLP and all umbilicals have been connected, workers will electrically and mechanically verify the mated interfaces to ensure all critical vehicle connections.

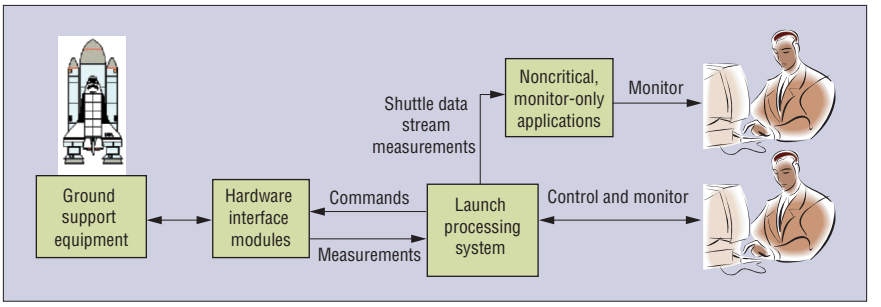


Figure 4. Ground control and monitoring at the NASA Kennedy Space Center.

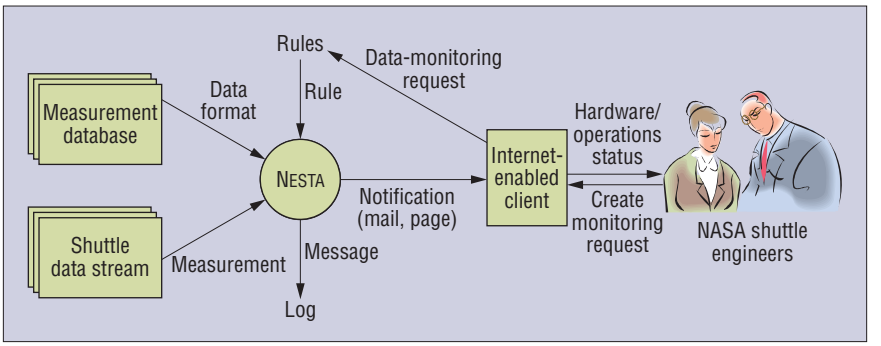


Figure 5. NESTA (NASA Engineering Shuttle Telemetry Agent) context diagram.

- Let users create and modify multiple monitoring requests without restarting NESTA.

High-level agent design

Figure 5 shows the context diagram for NESTA. The middle circle represents the agent process. NESTA communicates with various sources and data stores. A measurement database decodes the telemetry stream into usable data formats. The telemetry source broadcasts measurements as data packets over LANs. NESTA monitors the data stream for patterns specified by the shuttle engineers. The rules data store represents the scripts and knowledge base that define the monitoring criteria. If a pattern is matched, NESTA sends a notification as an email or wireless page. It also logs all messages and relevant agent activities locally.

We selected the Java Expert System Shell[2] as NESTA's rule engine. Because Jess was developed and supported at Sandia National Labs, another US government agency, the development team and customer have full use of the tool via government licensing without fees, including access to all of the Jess source code.

Jess modeled its forward-chaining reasoning system after production systems such as Clips and OPS5.[3,4] It contains highly efficient, sophisticated pattern matching based on the Rete algorithm,[5] enabling its inference engine to process many rules and data rapidly. The engine repeatedly processes through a match-select-act cycle. With NESTA, the consequent actions include sending notifications to shuttle engineers when the engine finds matches for predefined monitoring criteria. Jess also includes a fourth-generation scripting language and interactive command line that facilitate prototyping and testing.

Written entirely in Java, as is all of NESTA, Jess has access to the full Java API from the scripting language. The Jess API provides standard control flow constructs and supports variables, strings, objects, and function calls. Jess automatically converts between its own types and Java types, insulating the developer from performing the conversions manually. Used as a Java library, Jess supports multiple platforms through Java's "write once, run anywhere" paradigm. Beyond that, Java was a natural fit for NESTA, which has to support Web-enabled clients.

Agent deployment and payoff

NESTA is a peripheral advisory tool to LPS's real-time control system. NESTA's infusion of state-of-the-art AI technologies and engineering within the legacy launch system is particularly notable given the failures of several preceding attempts to modernize the ground-control system at KSC (see the "Why an AI Solution?" sidebar). Those failed projects had little to no spin-offs within the LPS community. In contrast, launch team members are accepting and internalizing NESTA. From a business vantage point, NESTA's greatest asset is its development and marketing as a value-added product. That helps pave its path to wider acceptance and use.

By December 2005, NESTA had been in use for two years, during which NASA engineers have written hundreds of rules and NESTA had generated thousands of notifications to multiple users at KSC and other remote sites. Because the customer is a NASA engineer responsible for oversight of contractors, the notifications act as an extra set of eyes that further assure the quality of government oversight.

NESTA's payoff has come in helping shuttle engineers meet their responsibilities, which include

- understanding their system and supporting equipment;

Why an AI Solution?

After the Shuttle Columbia disaster on 1 February 2003, the Columbia Accident Investigation Board proposed recommendations to improve safety from both organizational and technical perspectives.[1] The Board indicated the need to adopt "and maintain a shuttle flight schedule that is consistent with available resources." Also, both management and engineering support staff must maintain an awareness of anomalies and those must not be lost "as engineering risk analyses [move] through the process."

Given two tragic losses of a crew and shuttle, NASA engineers today are under greater pressure to be more vigilant in identifying problems. At the Kennedy Space Center, thousands of employees, both contractors and civil servants, perform shuttle ground processing. Anomalies must be detected and reported to prevent problems with shuttle subsystems, countdown, and launch. The aging Launch Processing System hardware has limited resources and precludes the level of automation and notification warranted by this domain.

NESTA leverages various AI technologies within a rule-based paradigm including forward chaining, fast pattern matching, declarative programming, and predicate logic. AI was a natural fit for monitoring shuttle ground-processing telemetry, as pattern recognition and analysis are the primary needs. Although regular expression libraries within various procedural and object-oriented languages can identify patterns, they aren't specifically intended for this type of application and have less efficient matching algorithms. The pattern-matching algorithms of rule-based expert system shells are highly specialized and tuned. Also, AI—particularly rule-based languages—lends itself better to this domain because pattern recognition wrapped within a premise-action construct closely mirrors the level of abstraction at which the domain experts work.

The type of data signatures sought by shuttle engineers requires the derivation of rules that are of the same granularity as those typically used in rule-based languages. Fortunately, shuttle engineers were already accustomed to representing knowledge at a fine-grained level. The engineers are adept at either constructing the rules themselves or expressing the knowledge in pseudocode that lends itself to translation directly into declarative rules. Many of the rules are either standalone or work in conjunction with several other rules. This suggests a highly modular system with a rule being a suitably sized working block.

The infusion of AI technologies, particularly rule-based libraries, has proved very fruitful. Interfacing and integrating these modern AI tools within a legacy launch system demonstrates the scalability and applicability of the tools and paradigm. The knowledge patterns that are evolving within NESTA will make it easier to train new users and also allow faster creation of rules.

Reference

1. H. Gehman et al., *Columbia Accident Investigation Board (CAIB)*, vol. 1, NASA, 2003.

- knowing how their systems are tested and processed;
- realizing when their systems are activated, tested, or in use;
- analyzing performance and data retrievals from any use of a system; and
- answering questions about their systems such as when was it tested, how did testing proceed, how did the data look, and is it ready to fly?

Let's look at three typical success stories.

Increased situational awareness

In one case, a shuttle avionics system powered up over a weekend. The shuttle engineer responsible for that system wouldn't have known that the system had powered up except for receiving a NESTA notification. In this case, the avionics user wasn't part of the engineer's immediate organization, so the engineer didn't receive any communiqués regarding the system's weekend usage. Because of NESTA, the Shuttle engineer was better prepared to address questions about his system's usage were they to arise.

This hasn't been an uncommon occurrence. NESTA users began realizing that some of the systems under their control were utilized much more than previously thought. This exemplifies the situational awareness that NESTA supports.

Increased efficiency

Some shuttle ground operations span 24 hours and include dozens of asynchronous events that are broadcast on the data stream. For example, checkout of flight control hardware in the orbiter processing facility occurred four to six times within the past year. The checkout included long hydraulic operations, powering up different parts of avionics, pressurizing and depressurizing the orbiter, and other work.

During a recent checkout, the NESTA notifications supplied exact times of events of interest to the shuttle engineer, providing a timestamped milestone history—a virtual roadmap—of lengthy testing operations that occurred while the engineer was performing other work. Before NESTA, the engineer had to glean the testing history by questioning various personnel or reviewing as-run test procedures and then guessing the time spans for retrieving selected measurement data. The research required to find the time parameters to enter into the data retrieval wasn't trivial, and the larger the time span on a data retrieval, the longer it took to retrieve it.

Just this one instance saved an hour of labor, letting the engineer focus on other concerns within this demanding real-time environment. The time-stamped roadmap eliminated the research time completely, and the individual retrievals were faster and smoother because the engineer knew the correct retrieval parameters immediately without having to guess.

Customer testimonial

Excerpts of an email received from a NESTA customer in April 2005 show how NESTA notified a NASA engineer of a hardware inspection that was not previously known to occur during a shuttle integrated test. We perform numerous tasks during this integrated test such as verifying the

NESTA Technical Challenges

As indicated within the high-level design and context diagram (see figure 5 in the main article), NESTA (NASA Engineering Shuttle Telemetry Agent) operates in a highly distributed environment: telemetry sources broadcast measurements as data packets over networks, notifications are generated by NESTA and sent to remote locations via a network, and so forth. This distributed intensive domain poses several technical challenges that must be addressed so that NESTA will not succumb to them.

Data validity

Because the shuttle data stream is published using the User Datagram Protocol, the connection is not always reliable and the network can drop packets. This poses problems when rules are waiting for data to arrive. Data health and validity become questionable. If the data stream connection gets lost entirely or the data becomes stale for lack of updates, false negatives or false positives can result, meaning that notifications of hardware events might never be sent or be sent in error.

To partially address this data validity issue, additional measurements are included in the rules to check for the stream's validity. Measurements are now marked invalid for dropped packets or when the source of the measurement becomes untrustworthy. There is still a larger problem of false negatives and never receiving an email if the data stream drops packets while a monitored event occurred. Aside from notifying the shuttle engineer of a data loss when it happens, we have not yet identified a mechanism that guarantees all notifications since the data stream is unreliable.

Measurement database changes

Multiple data streams and control rooms exist. Often, operations change the measurement database used to decode the Shuttle Data Stream. When that happens, decoding measurements becomes impossible and facts can no longer be updated in Jess's working memory. A short-term fix to this problem was to simply notify the NESTA system administrator when the stream changes. We added a measurement database Java bean, which is used within a user rule as a fact. When the measurement database changes, the administrator automatically gets an email and can restart NESTA accordingly. Longer-term, automatic restarts of the agent will be provided.

Email floods

If an end user incorrectly writes a rule, NESTA could flood the network and servers with hundreds or even thousands of notifications. To prevent that, we provided multiple safeguards, such as user-defined limits, to filter emails after NESTA has generated a given number for a particular email account.

Beyond that possibility of user error, a separate need exists to queue and merge multiple emails that might be related to some sequence. Queuing provides a mechanism for grouping multiple messages expected to occur within a short time period before emailing them in bulk. For example, four flight-control avionics boxes are often powered up in a short time period. Rather than a user receiving four separate and possibly interrelated flight control emails, we provided a queuing mechanism that lets a user tie related emails to the same queue and receive one bulk email compilation, rather than multiple emails. The end user can configure both the queue time and queue length.

interaction between the orbiter's avionics and the solid rocket booster's electro-hydraulic thrust vector control actuators.

> One of our NASA engineers came in for third shift Sunday to cover the testing. One important NASA function during this time period was star tracker light shade inspection. What happens in this test is that the star trackers are powered ON, the star tracker doors are opened, and then [the contractor] and NASA engineers inspect the inside of the star tracker—a cavity called a light shade which is a large cone coated with a black nonreflective coating and several baffles. The design of the light shade is to eliminate any and all extraneous light sources and reflections except for the star in view which the star tracker is trying to get a fix on. The inspection is made to make sure there is no foreign object debris. For example, a flake of paper could cause a reflection and lead to an erroneous star tracker star fix. If debris is found, special equipment is available to vacuum out the inside of the light shade. After this procedure, the star tracker is powered OFF and the star tracker door is closed for the last time at KSC.
>
> Now here's where NESTA paid off. During this third shift operation yesterday, [the contractor] and NASA were all on center waiting on the word from the S0008 test conductors to perform the star tracker light shade inspection. For whatever reason, our NASA engineer was never notified when the checkout was to begin. [The contractor] began the checkout without attempting to notify NASA. The first indication the NASA engineer had was when NESTA sent an email to the engineer announcing that the star tracker was powered ON. At this point, the NASA engineer contacted the test conductor and directed him to keep the doors open until he could witness the internal cavity inspection. Without NESTA, NASA would have missed the star tracker inspection. And this would have led to an uncomfortable discussion about whether the test would have to be repeated or whether NASA could rely solely on the eyes of the [contractor] engineers."

As is apparent, this testimonial demonstrates how NESTA helps track the complexity of an enormous number of testing events and their dependencies. The NESTA notification provided an increased awareness and might have prevented a further delay in testing of shuttle components. Humans might still "connect the dots" without the automated and reliable notifications coming from NESTA, but with NESTA, significant events are much less likely to be missed.

Monitoring launch commit criteria

NASA KSC has also used Jess to develop the Launch Commit Criteria Monitoring Agent.[6] In contrast to NESTA, which is used for day-to-day operations, LCCMA targets launch countdown activities.

During countdown, NASA shuttle engineers must monitor shuttle telemetry data for violations of launch commit criteria (LCC) and to verify that contractors troubleshoot problems correctly. When system engineers recognize a violation, they report it to the NASA test director. The report includes a description of the problem, the criticality, whether a hold is requested, and whether a preplanned troubleshooting procedure exists.

Shuttle engineers must monitor for many types of limit violations, ranging from simple high- and low-limit boundaries to much more complex first-order logic expressions. Each team has its own tools for identifying LCC violations. Many of these tools use the LPS software and simply change the displayed data's color or present a text message

to the user or set off an audible alarm. Troubleshooting might require other displays such as plots and troubleshooting flowcharts. Valuable time goes to locating these procedures and the data that supports them.

When LCCMA detects a launch commit criteria violation, it notifies the shuttle engineer through a workstation status display. Relevant troubleshooting procedures appear automatically on the display, ameliorating the need for the shuttle engineer to search for the correct procedure mapping to a given violation.

Beyond "Return to Flight"

The STS-114 "Return to Flight" collected unprecedented data on an orbiter's condition during ascent and in space. The mission tested new equipment, such as high-resolution cameras, and new procedures and conducted a first-of-its-kind spacewalking repair as well. Prior to the test flight, NASA put forward a significant effort to fix the foam problem that destroyed the Columbia orbiter and its crew. Unfortunately, the external tank again shed foam and NASA is once again tackling the problem. Shuttle flights are currently in stand-down mode until the problem is resolved.

The shuttle demands rigorous engineering to fix, maintain, and process its subsystems. The need to be forever diligent in managing its complexity continues, whether by redesigning shuttle subsystems or improving insight into ground-processing events. How to manage complexity and mitigate its adverse effects will also be emphasized in the engineering trade-offs conducted for future space vehicles as conceptualized within the Vision for Space Exploration.[7] These new space vehicles will represent best-of-breed engineering, a combination of carefully selected shuttle-derived components, and other proven technologies from space flight experience.

The "Technical Challenges" sidebar describes three hurdles that NESTA must surmount.

VSE presents new challenges. In particular, the need for autonomy significantly increases as people and payloads travel greater distances from Earth. Agents for these future applications will demand much higher degrees of autonomy than today's shuttle agents. Few or no human experts will reside at remote orbiting, lunar, or Martian sites to correct problems in a timely manner.

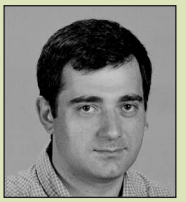
Glenn S. Semmel is a software lead within the Engineering Development directorate at NASA KSC, where he heads efforts to infuse AI-based technologies for ground support of the Space Shuttle and future space vehicles. Contact him at glenn.s.semmel@nasa.gov.

Daniel A. Rowe is a software and test engineer in the Engineering Development directorate at NASA KSC. Contact him at daniel.a.rowe@nasa.gov.

Steven R. Davis is a software engineer in the Engineering Development directorate at NASA KSC, with an interest in safety-critical control systems. Contact him at steve.davis@nasa.gov.

Kevin E. Smith is the lead flight control system test engineer within the Shuttle Processing Directorate at NASA KSC. Contact him at kevin.e.smith@nasa.gov.

Kurt W. Leucht is a software and test engineer in the Engineering Development directorate at NASA KSC. Contact him at kurt.leucht@nasa.gov.



Ladislau Bölöni is an assistant professor at the School of Electrical Engineering and Computer Science of the University of Central Florida. Contact him at lboloni@cs.ucf.edu.

Today on Earth, system and hardware engineers along with technicians leverage multiple skills when monitoring, diagnosing, and predicting problems in the Space Shuttle and its ground-support equipment. For VSE, the need for extending these skills to support other vehicles and payloads at remote spaceports ranging from the Earth to Mars becomes essential. These skills include the cognitive skills of being rational, collaborative, and goal driven, along with the ability to reason over time and uncertainty.

We are investigating extending NESTA to address some of the new VSE challenges. For example, an agent will need the ability to revise previously concluded assertions based on what now might be stale data. Also known as belief revision, this truth maintenance is particularly important when deep reasoning and extended inferencing is necessary.[8] An *assumption-based truth maintenance system* can reason over many contexts simultaneously. By capturing, maintaining, and deploying spaceport expertise within ATMS-enabled agents, we might reduce the costs and manpower required to meet the VSE goals, while increasing safety, reliability, and availability. ❐

References

1. *PCGOAL Requirements Document*, tech. report KSCL-1100-0804, Lockheed Space Operations Co., 1991.
2. E. Friedman-Hill, *Java Expert System Shell*, Manning Publications, 2003.
3. R.M. Wygant, "CLIPS: A Powerful Development and Delivery Expert System," *Computers and Industrial Eng.*, vol. 17, 2003, pp. 546–549.
4. L. Brownston et al., *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming*, Addison-Wesley, 1986.
5. C.L. Forgy, "Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem," *Artificial Intelligence*, vol. 19, no. 1, 1982, pp. 17–37.
6. G.S. Semmel et al., "Launch Commit Criteria Monitoring Agent," *Proc. 4th Int'l J. Conf. Autonomous Agents and Multiagent Systems* (AAMAS 2005), ACM Press, 2005, pp. 3–10.
7. *The Vision for Space Exploration*, tech report NP-2004-01-334-HQ, NASA, 2004.
8. J. Doyle, "A Truth Maintenance System," *Artificial Intelligence*, vol. 12, no. 3, Nov. 1979, pp. 231–272.