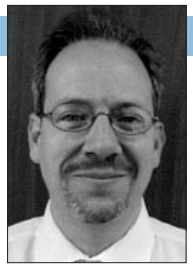


Richard Doyle
Jet Propulsion Lab
rdoyle@jpl.nasa.gov

Commanding and controlling satellite clusters

Paul Zetocha, Lance Self, Ross Wainwright, and Rich Burns, Air Force Research Laboratory
Margarita Brito and Derek Surka, Princeton Satellite Systems

Many organizations, including the National Aeronautics and Space Administration and the US Department of Defense, want to use constellations or fleets of autonomous spacecraft working together to accomplish complex mission objectives. At the Air Force Research Laboratory's Space Vehicles Directorate, we are developing architectures for commanding and controlling a cluster of cooperating satellites through autonomous software development for the TechSat 21 program (see the "TechSat 21" sidebar starting on page **XX**).

Many space missions require large, monolithic satellites. This often results in costly, complex, failure-prone vehicles whose physical size constraints limit their performance characteristics. Recently, various organizations have begun to explore how distributed clusters of cooperating satellites can replace their larger monolithic counterparts to reduce overall costs, enhance mission performance, and increase system fault tolerance (see Figure 1).[1]

Large clusters of satellites flying in formation must have some level of onboard autonomy to

- fly within specified tolerance levels;
- avoid collisions;
- address fault detection, isolation, and resolution (FDIR);
- share knowledge; and
- plan and schedule activities.

Commanding and controlling a large cluster of satellites can be very burdensome for ground operators as well.

This article describes our efforts to address these issues through the technology development for TechSat 21.

Cluster management

The AFRL is exploring technologies designed to allow a cluster of satellites to function as a single "virtual" satellite. The cluster-management technologies involve a combination of agent-based systems, the Spacecraft Command Language (SCL) from Interface and Control Systems, and Casper (Continuous Activity Scheduling Planning Execution and Replanning) software from NASA's Jet Propulsion Lab.

ObjectAgent. The ObjectAgent system is an agent-based, real-time architecture for distributed, autonomous control. In it, control systems decompose into agents, each of which is a multithreaded process. In these simulations, agents serve to implement all software functionality and communicate through a flexible messaging architecture (see Figure 2). Each message has a content field written in natural language that identifies the message's purpose and contents. Agents can load at any time and can configure themselves when launched, which simplifies the process of updating flight software and removes the complexity associated with software

Satellite clusters

A new class of space exploration mission is emerging that requires more than onboard decision-making capability. At NASA, these missions are sometimes called constellations, sometimes separated spacecraft. The defining attribute of this new mission class is that a mission is accomplished only through the use of multiple, not single, space platforms. Sometimes the platforms are all of a type, and the challenge is to fly them in a precise formation, for example to synthesize a large observing aperture from components. But a constellation can consist of heterogeneous assets as well. A fleet of Earth-observing satellites with different sensors and instruments, all working together to monitor hazards such as volcanic eruptions and forest fires, is one example. At Mars, the goal is to ultimately have both orbital and landed assets (rovers and permanent science stations) in place, and perhaps even airborne assets, all cooperating to perform scientific investigations of the Mars environment.

Separated spacecraft missions will require a layer of middleware to manage distributed space assets. The distributed character extends to all spacecraft functions (guidance, navigation and control, fault protection, resource management and mission planning) and extends the existing challenge to create autonomous space systems in a new direction.

The authors describe work on creating distributed autonomy infrastructure for coordinated Earth-orbiting space assets. In this case, the mission context is not NASA, but the space efforts of the US Air Force.

—Richard Doyle

patches. They will automatically seek out other agents that can give them the inputs they need. Decision-making, fault detection, and recovery capabilities are also built in at all levels.[2]

Agents in ObjectAgent work at all levels of software functionality because there is no set level of complexity for an agent. For instance, a designer could choose to implement the entire flight software as a single agent, use one agent for the software related to each subsystem, or use multiple agents for each subsystem. ObjectAgent agents are composed of skills. An agent's skills determine its complexity and functionality. However, all agents have some basic skills to ensure that they can communicate. In addition, agents have self-knowledge and can explain their functioning and purpose to other agents and users. Agent communication occurs solely through messages: there is no shared memory between agents. This approach ensures that agents can work together even when they are not located on the same processor.[2]

The ObjectAgent software package provides a GUI-based development environment for designing and simulating multiagent systems. This design environment simplifies the agent-creation process and provides a common interface to a number of advanced control and estimation techniques. Figure 3 shows the Agent Developer GUI.

TeamAgent. The TeamAgent system uses ObjectAgent for commanding and controlling multiple cooperative satellites. TeamAgent enables agent-based multisatellite systems to fulfill complex mission objectives by autonomously making high- and low-level decisions based on the information available to any of the satellite system's agents. Princeton Satellite Systems has identified the required spacecraft functions for multiple spacecraft missions and has demonstrated the use of software agents and multiagent-based organizations to satisfy these functions. Furthermore, we have used TeamAgent in developing multiagent system simulations for multiple satellites to illustrate collision avoidance and reconfiguration for a four-satellite constellation. The agents monitor for collisions, reconfigure the fleet, optimize fuel usage across the cluster during reconfiguration, and develop a fuel-optimal maneuver for reconfiguration.

During the development's first phase, we prototyped ObjectAgent and TeamAgent in Matlab and developed a complete, GUI-based environment for creating, simulating, and analyzing multiagent, multisatellite systems. Currently, we are porting ObjectAgent and TeamAgent from Matlab to C++ for implementation on a real-time system.

We will use ObjectAgent and TeamAgent to build two flight software elements: the cluster manager and the spacecraft manager. The cluster manager performs relative control of the cluster's satellites, including relative stationkeeping and estimation of the cluster's center-of-mass and each satellite's relative positions, as well as cluster-level commanding, health summarization, and fault detection. The spacecraft manager performs many functions that would normally reside on the ground, including spacecraft-level fault detection. It

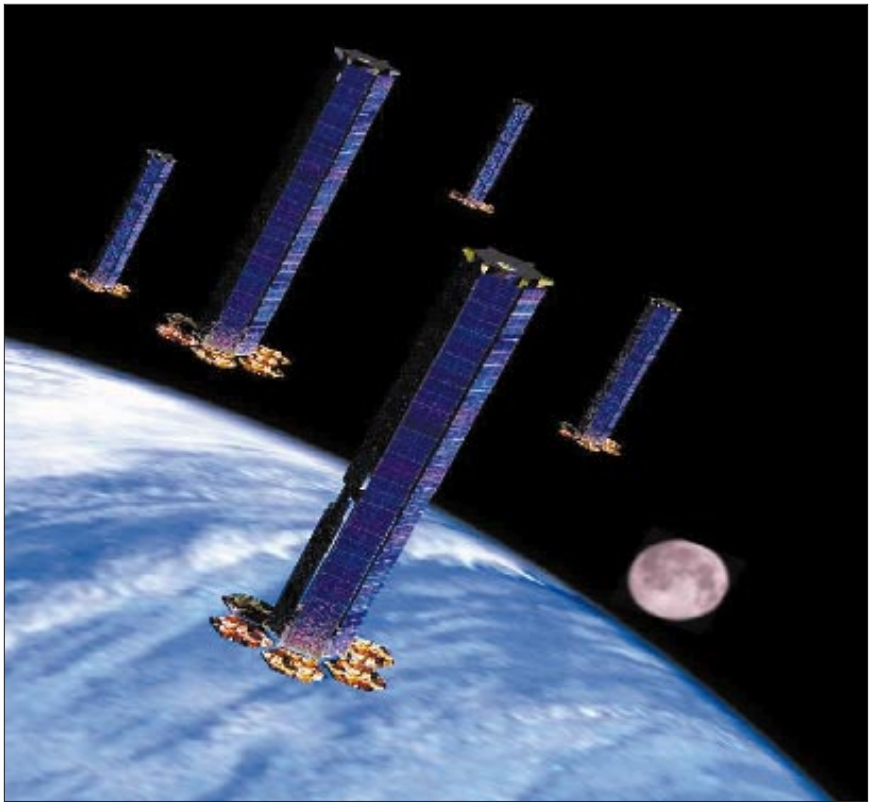


Figure 1. Artist's rendition of satellites flying in formation.

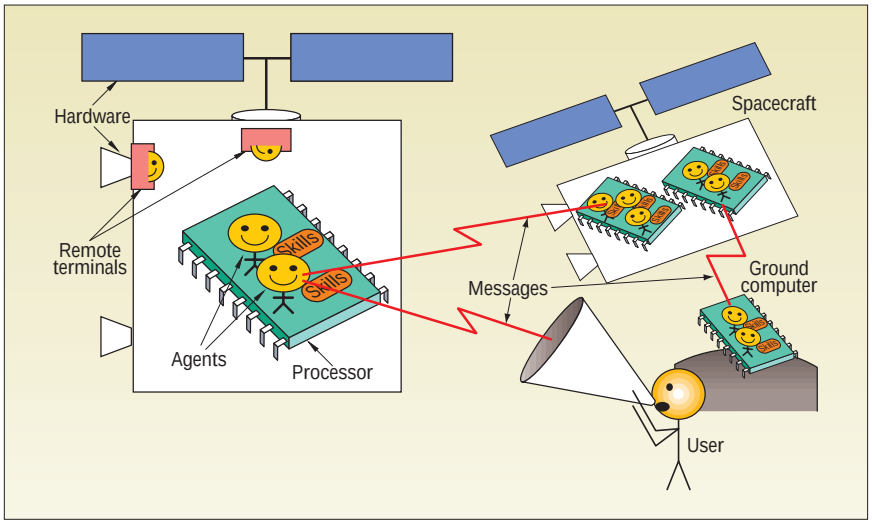


Figure 2. Agent messaging system.

TechSat 21

The Air Force Research Laboratory (AFRL) has initiated the TechSat 21 program to serve as a proof-of-concept mission for a new design paradigm for space missions. This paradigm seeks to reduce costs and increase system robustness and maintainability by distributing functionality over several microsatellites flying in formation. The distributed functionality includes processing, communications, and control functions, as well as payload functions. Thus, the system of microsatellites forms a virtual satellite, which its operators can control and task as a single satellite.[1]

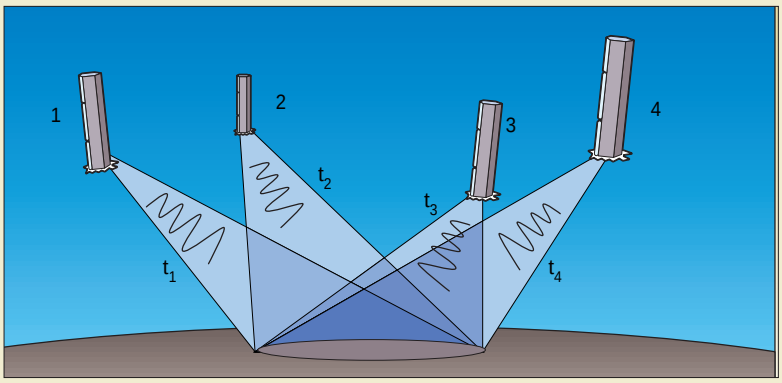


Figure A. Sparse aperture array.

Spurred on by the potential of reduced launch costs, increased system robustness, and enhanced maintainability, the topic of formation flying has attracted considerable interest. Several formation flying missions are planned, including NASA's Earth Orbiter 1 and Space Technology 3 and 5, the University Nanosatellite Program, Laser Interferometer Space Antenna (LISA), and Discoverer II. Among these systems, TechSat 21, which has several mission objectives, has some of the most stringent requirements for formation flying given its interferometric-based mission in low-Earth orbit. TechSat 21 will perform a distributed sparse aperture radar mission for ground-moving target indication and geolocation missions, which will let us compare this concept's performance to single-satellite systems. This approach also addresses missions of significant technological challenge, including heavy onboard processing requirements, tight relative navigation accuracy, significant satellite-to-satellite and satellite-to-ground communications loads, and precision formation control on tight fuel budgets, among others.[2]

Formation flying

Control of the TechSat 21 has a nominal requirement to maintain the

manages the spacecraft flight software, while the cluster manager manages the spacecraft managers.

Princeton Satellite Systems is developing the ObjectAgent and TeamAgent systems under two Phase II SBIR contracts from the AFRL's Space Vehicles Directorate.

Cluster manager

Cluster manager functionality falls into four major areas:

- command and control,
- cluster data management,
- formation flying, and
- fault management.

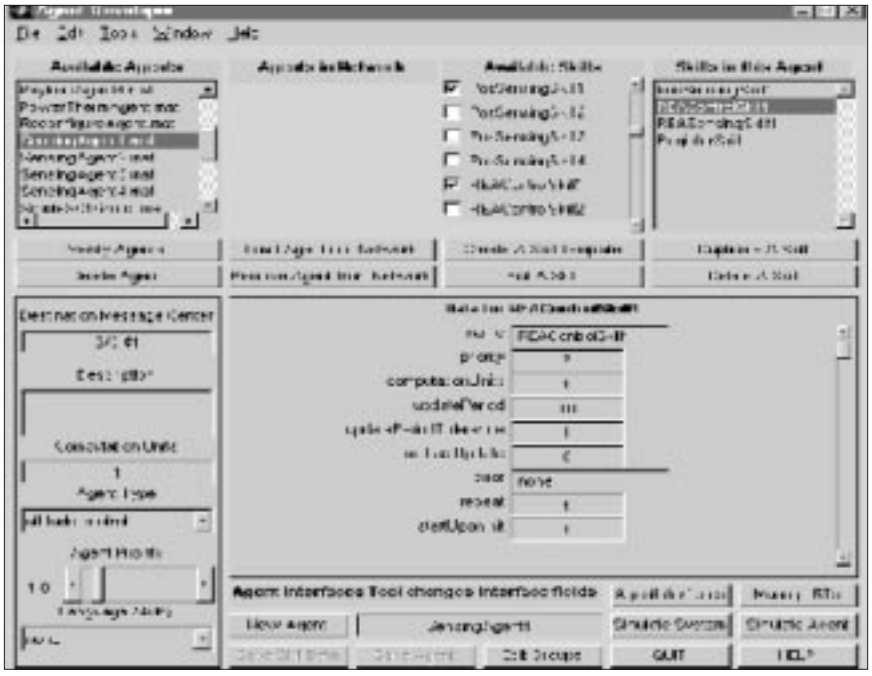


Figure 3. The ObjectAgent GUI lets users create and configure agents.

We used a combination of SCL, Casper, and ObjectAgent to implement the cluster manager's command and control. SCL alone served for implementing intersatellite communication and ground–cluster-manager communication. Casper helps break down high-level commands into lower-level commands and plan implementation of complex tasks. Both SCL and ObjectAgent serve to generate commands for other spacecraft, depending on the type of algorithm generating the command. The cluster manager's command and control capabilities are what let us treat the cluster as a virtual satellite.[3]

Because the cluster must provide state-of-health information for all its satellites, we needed to develop cluster data management. It must also keep track of data, such as relative position and velocity, needed to control the cluster. Potentially, the cluster must be able to provide any telemetry data requested by the ground for any of its satellites. The SCL database keeps track of all necessary data. The satellites provide some data to the cluster manager periodically and other data only on request. We are still determining what information the cluster manager should keep in its database, but at a minimum it includes the following (for each satellite in the cluster):

- relative position,
- relative velocity,
- absolute position,
- absolute velocity,

separations within the formation to approximately 10% of their values. Formations with separations on the order of 1 km will be controlled to 100 m, while for separations on the order of 100 m, separation-distance control will be to 10 m—a truly staggering requirement when you consider the problem's scale. The TechSat-21 orbits will be nearly circular at altitudes near 600 km, so controlling to this level of precision will require minute corrections to the orbital elements of the formation's individual members affected by appropriately small thrusts.[2]

The TechSat 21 mission calls for the collection and combination of signals irradiated from each satellite and reflected from the target (Figure A). Therefore, we must know the relative positions of the formation's members to the millimeter level. Because the signals themselves can serve for this relative-position determination, we can relax the requirement for the relative navigation system using measurements external to the radar signals to the centimeter level.[2]

Distributed sparse aperture radar

TechSat 21 will assess the utility of the space-based, sparse-array aperture formed by the satellite cluster. For TechSat 21, the sparse array will serve to synthesize a large radar antenna. TechSat-21 will perform a distributed sparse aperture radar mission for ground moving target indication and geolocation missions, which will provide a means for comparison to single satellite systems.

The cluster is a dynamic array, which is both an advantage and a disadvantage for the TechSat concept. The angular resolution of two smaller antennae working together is the same as one large antenna with a diameter equal to the separation of the two small antennae. Skillful positioning of the elements of the TechSat cluster can optimize the sparse-array for a particular application. But, position and timing uncertainties inherent in separately orbiting objects can limit performance. AFRL researchers are working on methods to ensure that position and timing requirements are met by using the global positioning system and intersatellite links, among other techniques.

While many small antennae are theoretically the same as one large antenna, in fact, smaller antennae create large spot sizes resulting in range and Doppler ambiguities and, ultimately, blurry images. AFRL researchers are investigating innovative processing techniques, optimized waveforms, and the use of multiple waveforms to improve range and Doppler resolution.

References

1. R. Burns et al., "TechSat 21: Formation Design, Control, and Simulation," *Proc. IEEE Aerospace 2001 Conf.*," IEEE Press, Piscataway, N.J., to be published in 2001.
2. A. Das, R. Cobb, and M. Stallard, *TechSat 21: A Revolutionary Concept in Distributed Space Based Sensing*, Am. Inst. Aeronautics and Astronautics, Washington, D.C., AIAA 98-5255, 1998

- attitude quaternion,
- system time,
- spacecraft mode,
- fuel level,
- reference trajectory, and
- sensor states.

The cluster manager's formation-flying component maintains the cluster formation and reconfigures the cluster whenever necessary. We have implemented the algorithms necessary for formation flying as ObjectAgent agents. The inputs needed by the agents sometimes come from other agents and sometimes from the SCL database. The outputs from the cluster manager's formation-flying segment are commands for the cluster members. Thus, we have implemented the formation-flying portion using ObjectAgent, but it has strong interfaces to both the cluster data-management and command-and-control portions.

The cluster manager will be responsible for identifying and handling cluster-level faults. These are faults that require action from the cluster to be managed—for example, a failure of one satellite's intersatellite link. In this case, although the spacecraft in question might still be able to function as an individual satellite, it can no longer participate as a member of the cluster. The cluster manager must compensate for that fact. A combination of ObjectAgent and SCL will implement cluster-level fault management.

Ground system

Commanding and controlling a cluster of satellites from the ground poses many challenges. Optimizing ground operation cost and manpower requires different methods to command the cluster, monitor it, and perform telemetry decommutation. Because the ground station must interact with the virtual satellite, we are taking a systems-level approach, which means that we will use the SCL both on the ground and in space, giving SCL an especially important role in our cluster-management system.

For a large cluster, it is not efficient or cost effective to command satellites individually. We can more efficiently send commands to the onboard cluster manager and have the cluster manager either parse the command string and forward the commands to the appropriate satellites or make some intelligent decision as to the most appropriate action. Two types of ground commanding are possible. The first type involves sending up a sequence of commands that are intended for specific satellites. This command string would go to the onboard cluster manager, which would then parse the command string and send commands to the appropriate satellites.[4]

A second type of commanding involves commands that go to the cluster without specifically indicating which satellites in the cluster will ultimately be affected. A hypothetical example might be to issue a command to "observe region x at time y." The cluster manager, based on the status of the satellites at time y, will then determine the appropriate course of action. This type of commanding requires more onboard intelligence than the first scenario. It also entails a higher level of risk, so safeguards must be put in place to ensure that no adverse conditions arise.

Telemetry decommutation on the ground requires that we be able to parse telemetry from multiple satellites. For a TDM-based telemetry system, this scales nicely from how traditional TDM-based systems oper-

Terms

AFRL	Air Force Research Laboratory
Casper	Continuous activity scheduling planning execution and replanning software
CCSDS	Consultative Committee for Space Data Systems
C&DH	Command and data handling
COTS	Commercial off-the-shelf
FDIR	Fault detection, isolation, and resolution
PSS	Princeton Satellite Systems
SBIR	Small Business Innovation Research program
SCL	Spacecraft Command Language
SQL	Structured Query Language
TDM	Time division multiplexer

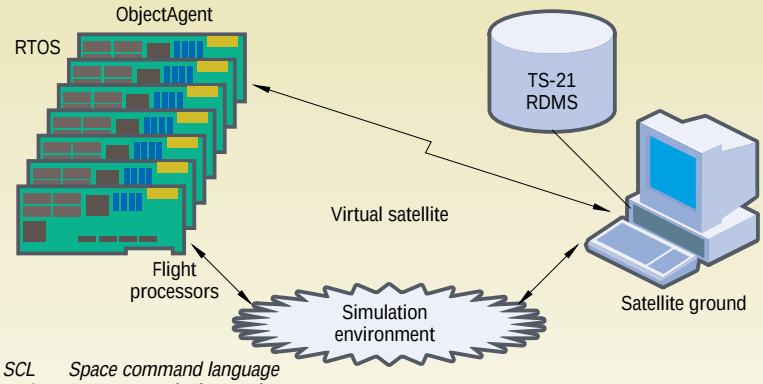


Figure 4. Ground station GUI.

ate. Telemetry from different satellites simply gets associated with specified frames and frame locations. For CCSDS-based systems, packets can contain a field that identifies where they originated.

Developing methods to monitor and visualize the cluster's state of health can be challenging. Visualizing individual satellite mnemonics can be difficult for moderately complex satellites—a problem that gets magnified for a cluster of satellites. One solution is to develop a hierarchical telemetry-display system. A top-level system would create the overall status of individual satellites. Choosing an individual satellite and drilling down to the second level would contain a display showing all subsystems for that satellite. Additional levels would partition a subsystem even further. Anomalous conditions would be highlighted at any level by bubbling up problems through the hierarchy.

As the core of our ground system, we are baselining the SCL, a commercial-off-the-shelf (COTS) software package that contains an expert system and a command scripting language. Designed to operate both onboard a satellite and on the ground, it is an ideal environment for developing a system that contains the cluster commanding and monitoring capability we described earlier. Expert system rules can be developed and migrated from ground to space, as appropriate. Using the rule-based expert system, we can develop a fault tree for known anomalous conditions. Figure 4 shows an initial prototype of a ground station GUI, which provides cluster-level state-of-health monitoring and control. We are developing a prototype that adds the hierarchical monitoring capability.

SCL will also handle all commanding and telemetry between the ground and the cluster and simulate the onboard Command and Data Handling (C&DH) system.

TechSat 21 testbed. We are developing a distributed satellite testbed to develop and test the various satellite cluster command and control technologies, as well as other TechSat 21 technologies. Figure 5 shows a simplified version of the TechSat 21 testbed.[3]

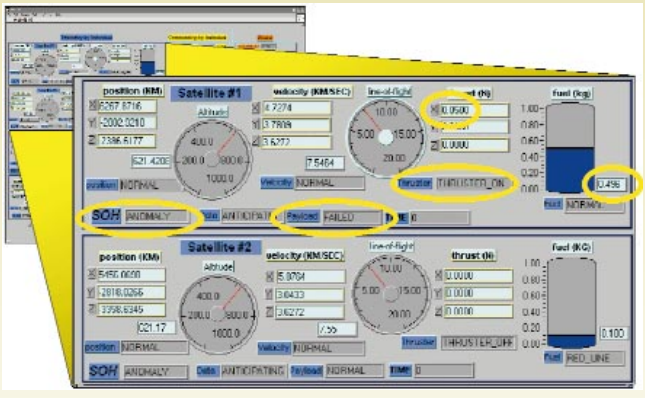


Figure 5. TechSat 21 testbed.

The testbed's flight system section consists of eight Force PowerCore 6750 boards, each having a single PowerPC 750 processor and housed in a VME chassis. They are connected using 100-Mbps Ethernet. In addition, each board has two RS-232 interfaces. Each board is running Enea's OSE real-time operating system, a message-passing OS well suited for distributed applications.

As Figure 5 shows, the flight system interfaces with a simulation environment and the testbed's ground segment. The simulation includes spacecraft dynamics, environmental factors, and actuator and sensor models. It provides inputs to the software on the flight boards and receives software outputs to the spacecraft actuators. Currently, the simulation connects to each board through one of its serial interfaces. In the future, the boards will interface with the simulation environment through Ethernet. SCL, currently at both ends of the interface, handles communications between the ground and flight systems through the Ethernet.

The testbed can simulate a cluster of up to eight satellites with the flight software for a single satellite running on each processing board. The prototype system being testing now, however, consists of a three-satellite cluster organized in a leader–-follower fashion. The cluster manager, or leader satellite, carries software to let it make cluster-level decisions and issue commands to the follower satellites. Because this system's two follower satellites are designated as primary and secondary backups to the cluster manager, they carry the same software onboard; however, this software stays turned off until the satellite needs to assume the function of cluster manager.

The simulation environment contains a wide variety of modules necessary to do a complete end-to-end simulation, including environment, target, orbit determination and control, and payload modules. Raw telemetry data resides in an SQL database and can be accessed remotely via HTML.

Testbed data center. In support of the testbed and TechSat 21, the AFRL is developing a data center, which will support these efforts and the customer base that will exploit the experimental results. Data center customers range from mission planners and

system engineers to payload analysts and several other specialties, including satellite engineers and analysts. This diverse customer base has needs ranging from information that lets them make high-level planning and scheduling decisions to raw data for prediction of satellite subsystem performance. All data center customers, however, share a common need for a stable, secure, and accessible environment for recreating and exploiting irreplaceable data sets. To satisfy these diverse requirements, the data center will be a central point to manage, secure, and distribute the data sets and provide the following basic services:

- manage data access for customers within and external to the AFRL;
- provide a central archive site for payload, satellite, targets, documentation, and modeling and simulation data sets;
- provide data access via the Internet; and
- provide and maintain utility programs for archiving, cataloging, processing, replaying, and accessing data sets.

The technologies we're developing could well satisfy the command and control of a cluster of satellites with virtually no cost increase over their monolithic counterparts. We plan to flight test our cluster manager onboard TechSat-21, which is scheduled for launch in early 2004. If successful, this could enable the command, control, and maintenance of a whole family of next-generation spacecraft clusters and constellations. ■

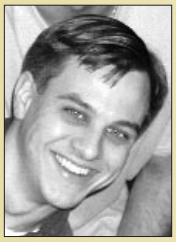
Acknowledgment

Much of the cluster-management research described here is being accomplished by Princeton Satellite Systems and Interface and Control Systems under Small Business Innovative Research contracts and managed by the Space Vehicles Directorate of AFRL. The distributed satellite testbed is a resource of AFRL/VSSW and is located at Kirtland Air Force Base, Albuquerque, New Mexico.

References

1. P. Zetocha, "Intelligent Agent Architecture for Onboard Executive Satellite Control," *Intelligent Automation and Control*, vol. 9, TSI Press Series on Intelligent Automation and Soft Computing, Albuquerque, NM, pp. 27–32, 2000.
2. D.M. Surka, M.C. Brito, and C.G. Harvey, "Development of the Real-Time Object-Agent Flight Software Architecture for Distributed Satellite Systems," *IEEE Aerospace Conf.*, IEEE Press, Piscataway, N.J., to be published in 2001.
3. P. Zetocha and M. Brito, "Development of a Testbed for Distributed Satellite Command and Control," *Proc. IEEE Aerospace Conf.*, IEEE Press, Piscataway, N.J., to be published in 2001.
4. P. Zetocha, "Satellite Cluster Command and Control," *Proc. IEEE Aerospace Conf.*, IEEE Press, Piscataway, N.J., 2000, CD-ROM.

Paul Zetocha is a computer engineer, currently leading the Intelligent Satellite Systems Group of the Air Force Research Laboratory's Space Vehicles Directorate. He is also the chairman of the AIAA Intelligent Systems Technical Committee. He received MS degrees in electrical engineering and computer science from the University of New Mexico with an emphasis in signal processing and AI. He is a senior member of AIAA and a member of IEEE. Contact him at the Air Force Research Lab, AFRL/VSSW, Kirtland AFB NM, 87117; paul.zetocha@kirtland.af.mil

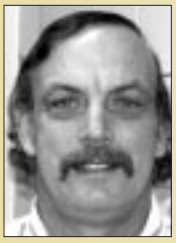


Derek Surka is a member of the Technical Staff at Princeton Satellite Systems. He is the principal investigator on TeamAgent and has worked in the areas of spacecraft autonomy and control since 1994. He received his BS in aerospace engineering from Caltech and an SM in aerospace engineering from MIT and is an avid curler. Mr Surka is a member of AIAA. Contact him at Princeton Satellite Systems, 33 Witherspoon St., Princeton, NJ 08542; dmsurka@psatellite.com

Margarita Brito is an aerospace engineer with Princeton Satellite Systems. She is working with others to develop ObjectAgent software to run on the OSE Real Time Operating System. In addition, she is responsible for the integration of ObjectAgent software into the AFRL TechSat 21 Testbed. She holds an MS in aerospace engineering from MIT. Contact her at Princeton Satellite Systems, 33 Witherspoon St, Princeton, NJ 08542; megui@psatellite.com

Rich Burns is a research engineer at the AFRL specializing in formation flying, satellite theory, and nonlinear dynamic systems. He received his MS from Stanford University and his BS from the University of Notre Dame, both in Mechanical Engineering. He is a member of AIAA and is a member of the Astrodynamics Technical Committee. Contact him at the Air Force Research Lab, AFRL/VSSW, Kirtland AFB NM, 87117; rich.burns@kirtland.af.mil

Ross Wainwright is a computer scientist at the AFRL working in the Intelligent Satellite Systems Group. He has been involved in several satellite programs dealing with automation of ground operations. He received a BS from the University of California, Davis and an MS from California State University, Los Angeles, both in applied mathematics. He is a member of the ACM. Contact him at the Air Force Research Lab, AFRL/VSSW, Kirtland AFB NM, 87117; ross.wainwright@kirtland.af.mil.



Lance Self is a computer scientist at the AFRL working in the Intelligent Satellite Systems Group. He received his BS in mathematics from the University of Hawaii and his MS in engineering management from West Coast University. He has worked on several missile test programs for the US Navy and currently heads up the AFRL effort to create a data center to support the TechSat 21 satellite program. He is a member of the ACM. Contact him at the Air Force Research Lab, AFRL/VSSW, Kirtland AFB, NM, 87117; lance.self@kirtland.af.mil