Space station hands NASA its biggest software task yet

John Wolfsberger, NASA
Marshall Space Flight Center

To create a permanent, manned space station, the National Aeronautics and Space Administration will divide the development effort among four centers, requiring the agency to manage a distributed project larger than anything it has ever attempted to handle. NASA engineers must integrate and test the software systems from several independent, widely separated designers and perform long-term activities over as many as 30 years.

The project’s size and complexity prompted a NASA software working group to recommend that:

- A consistent software management approach be developed and applied to all software-contributing participants.
- Software standards and policies be established, implemented, and enforced throughout the program.
- NASA establish a common software-support environment that everyone can use to produce customized software.
- One language — Ada — be designated for all space station application software.

NASA headquarters has since implemented the recommendations by letting a contract for a technical and management information system (TMIS) to Boeing and a contract for a software-support environment (SSE) to Lockheed. The TMIS will enforce consistent management and documentation, while the SSE will enforce consistent tool use, simulation techniques, and interfaces.

Management. The space agency has approved software-management policies intended to ensure top-level policy direction and common software policies. The policies address a standard software life-cycle model specifying the phases of software acquisition, associated reviews, documentation policies, and standard terminologies.

Key software development will occur at the Marshall Space Flight Center in Huntsville, Ala., the Johnson Space Center in Houston, the Goddard Space Flight Center in Greenbelt, Md., and the Lewis Research Center in Cleveland, Ohio. International partners — Canada, Japan, and the European Space Agency — will also develop secondary hardware and software systems.

NASA headquarters in Washington, DC, is ultimately responsible for enforcing standards and integrating the system pieces (including simulators) supplied by the various sites. To help do this, software components, algorithms, and simulation packages will be kept in libraries available to all centers, ensuring consistency and trustworthiness as well as minimizing replication. However, NASA engineers must yet establish a methodology to ensure the consistency and quality of library components.

The Marshall center has begun planning and implementing a software-management approach to accomplish its assignment. While the project’s scope is larger than previous NASA efforts, Marshall software managers are following techniques from current and past programs, including Skylab and the space shuttle.

Based on updated versions of the 1979 benchmark “MSFC Software Management and Development Requirements” document (MA-001-006-2H) and consistent with the overall software-management policy document, Marshall project managers have developed a software-management plan that establishes policies, procedures, and guidelines to be used during all development phases.

The Marshall team has begun planning for system integration and testing. It already has plans for verification and validation of software integration of the habitation, laboratory, and logistics modules. Because these elements will include logic-software packages provided by other centers, the Marshall software manager is developing plans to integrate and test software provided by other centers.

Design. The software will contain not only the unique flight-software packages required to operate the station but also many services that in the past were found in ground-based software. The on-board and ground systems will rely on custom-built software, commercial off-the-shelf software, and firmware packages. Decisions have yet to be made about the systems’ architectures, but several prototype architectures are being tested. Regardless of the configuration, the system will require a high degree of reliability while performing real-time and near-real-time functions and sustaining operations 24 hours a day.

Considering the station’s potential 30-year life span and the rate at which technology is moving, keeping the space station software current presents a set of unique challenges for the systems manager. Initial planning must let capabilities under development and those not yet thought of be added without disrupting the station’s operational capabilities. This is one of the biggest challenges NASA will face, and NASA must develop a methodology to handle the insertion of new technology over the next 30 years.

Expert systems may play a role in the space station software, but many questions remain about their trustworthiness. One approach under consideration is to implement expert systems in parallel to other systems (but off-line) and compare their decisions to those made by NASA engineers. If the expert systems make appropriate decisions, they could be made part of the actual system. The key question will be determining if these expert systems really work before using them to automate the station or parts of it.

Quality assurance. The program’s hierarchical management, standards enforcement, common work approaches, and use of library components are designed to ensure that defects are caught before they become critical. Simulation at several levels will help verify quality. Packages will be tested independently, as related groups, as groups of groups, and ultimately as the entire system. This system simulation will most likely continue to be used after the space station itself is functioning so NASA can test changes, upgrades, and fixes before applying them to the real station. NASA has used this technique successfully since the Apollo missions to the Moon.

Wolfsberger is lead engineer for new software technology at the Marshall Information and Electronics Data Systems Laboratory. He has worked with NASA since the Apollo missions.

Correction

The September issue’s report on software-engineering programs (pp. 88, 90) mistakenly said that Texas Christian University’s software-engineering program no longer exists. The graduate program has run continuously since 1978, although it changed its name in 1981 from software engineering to software design and development, said James Comer, the university’s Computer Science Dept. chairman. The program offers six courses to about 50 students, most of whom work for local industry, he said.