Technology | DOI:10.1145/2408776.2408782 Alex Wright

Revving the Rover

The new Mars rover has attracted plenty of attention for its planetary gymnastics, but the big breakthroughs are under the hood.

When NASA's rover Curiosity touched down on the surface of the red planet on August 6, 2012, the cheering in Mission Control was soon echoed by a prolonged burst of public euphoria. Crowds gathered in Times Square at one in the morning to watch the landing on a giant TV screen, while millions of Web users blogged, tweeted, and otherwise applauded the embattled space agency's continuing ability to pull off Big Things.

Much of the Internet chatter centered on the acrobatic Sky Crane maneuver, in which the landing capsule morphed like a Transformer into a rocket-powered hovercraft to ease its precious cargo onto the planet surface. While the landing made for great online theater, those arresting images may also have diverted attention from several other important, though admittedly less telegenic innovations.

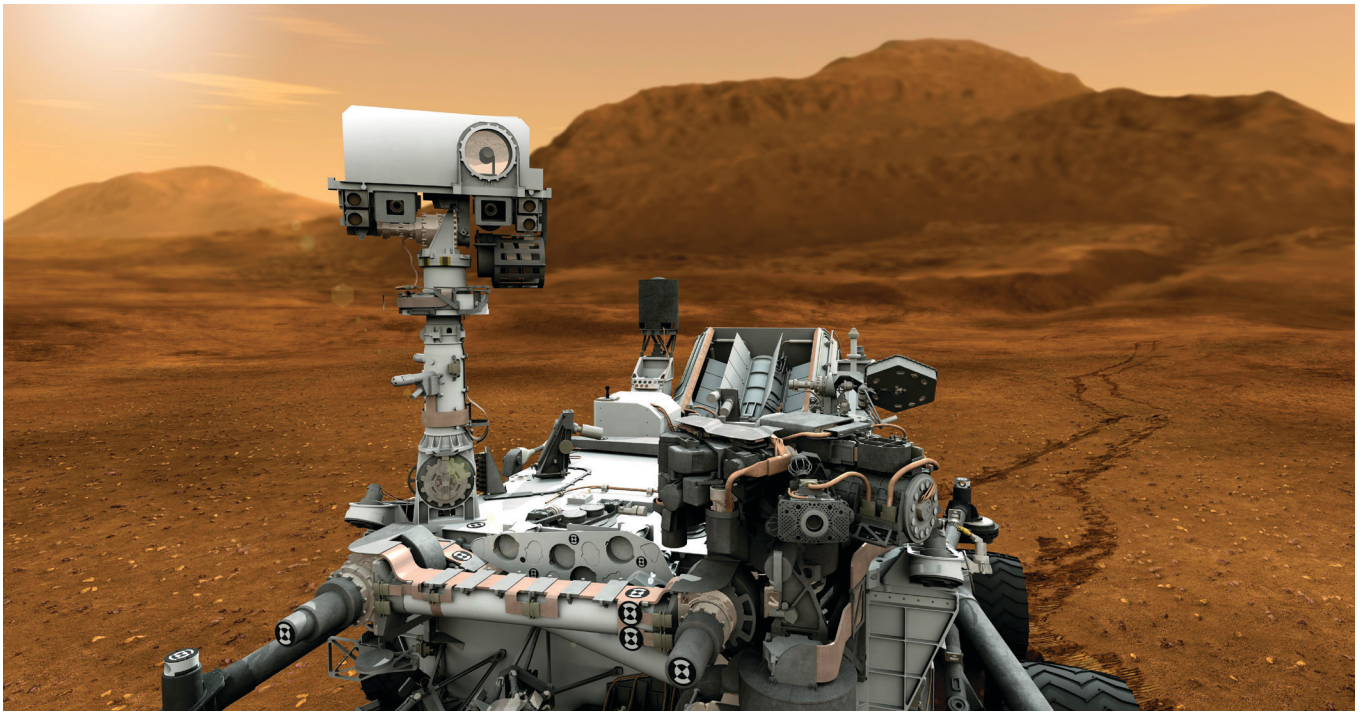
The Curiosity mission (formally known as the Mars Science Laboratory, or MSL) may also mark the end of an era for NASA, as planetary exploration approaches a level of engineering complexity that may call for fundamentally rethinking the design and architecture of future robotic missions.

"This represents the arc of an engineering process that really started in the 1960s," says Rob Manning, chief engineer of the Mars Exploration Program at NASA's Jet Propulsion Laboratory in Pasadena, California.

The earliest NASA missions of the 1960s and 1970s relied on highly distributed systems, with computing power resident on multiple devices, largely due to limitations in processing power. Starting with the Mars Pathfinder mission in 1996, however, the agency started to embrace a more centralized model, concentrating most computing tasks onto a single onboard computer.

While the basic contours of each rover mission have stayed roughly the same since then—namely, fly a spaceship to Mars, land a wheeled vehicle, then collect data while driving around the planet surface—the data gathering requirements have grown progressively more sophisticated with each mission.

The 1996 Sojourner rover was content to snap photos and perform x-ray spectrometry on a few rock samples within about 40 feet of the landing site. For the 2004 mission, the team gave the Spirit and Opportunity rovers considerably more autonomy, equipping them with a new software system dubbed Autonomous Exploration for Gathering Increased Science (AEGIS) that allowed the rovers to select potentially interesting research targets without requiring direction from Earth-bound controllers. Curiosity takes that autonomy several steps further, moving far and wide—powered by a plutonium-fueled nuclear engine—



NASA's rover Curiosity landed on Mars last August carrying almost 2,000 pounds of state-of-the-art scientific instruments.

IMAGE COURTESY OF NASA/JPL-CALTECH

while carrying an arsenal of 10 different scientific instruments, including cameras and imaging equipment, environmental sensors, and sophisticated sampling tools. Curiosity weighs in at nearly 2,000 pounds (compared to Sojourner's lithe 23 pounds).

Managing so much onboard equipment constituted an enormously difficult hardware and software design challenge. As the scientific requirements have grown more elaborate, the team has discovered the downside of centralized computing.

"We have one set of requirements for cruise, one for landing, one for on the surface," explains Manning. "So we have all this extra hardware and interfaces—and now we have to lug it all around."

While the all-in-one approach makes for a much bulkier machine than previous rovers, the complexity of the software stems primarily from the rover's high degree of autonomy, demanding millions of lines of code that would allow the rover to navigate the planet surface, identify and react to potential hazards while collecting samples, aim precision-targeted laser beams onto rocks several meters away, and communicate with Earth via its interplanetary ISP, NASA's prodigious Deep Space Network.

Managing so many discrete functions on the same machine demands a high level of functional decomposition, so that different routines can take over the system at appropriate times without compromising other essential features. As a result, the engineering team had to think carefully about issues of memory allocation and fault tolerance, as well as managing a bewildering array of input and output devices.

"Software grows exponentially fast in this domain," says Gerard Holzmann, head of the Laboratory for Reliable Software at NASA's Jet Propulsion Laboratory, an organization formed in 2003 to improve the reliability of mission-critical software. Indeed, with each successive mission to Mars, the size of the onboard flight code has more than doubled. While software engineers have long understood that software packages often grow over time—expanding to take advantage of faster processors and additional storage space, for example—that problem gets further exacerbated by the complexities of safety-critical systems like space exploration.

The Curiosity mission may mark the end of an era for NASA.

"Managing the development of a few million lines of critical code carries very different challenges from the development of a few thousand or even a few hundred thousand lines," says Holzmann.

To cope with the scale of the MSL challenge, the team introduced several important new software reliability initiatives, including the design of a new Institutional Coding Standard that, while requiring relatively few strict rules, was designed to support automated compliance verification, allowing the team to run a nightly check on every new build. The team also introduced a new peer code review process and "scrub" tool integrated with a suite of static source code analysis tools including well-known commercial testing and analysis tools like Coverity, Codesonar, and Semmle, as well as Uno, a source code analysis tool that Holzmann had developed several years earlier while working at Bell Labs.

The static source code analyzers played a critical role in the software development process, allowing the team to ferret out risks of data corruption, race conditions, or deadlocks.

"A good static analyzer is very much like employing an additional, very conscientious and tireless developer on your team," says Holzmann. "It's an extra set of eyes that never tires of pointing out new subtle flaws."

The team also had a secret weapon on hand in the form of Holzmann's Spin logic model checker, which he developed over a period of several decades in his previous job at Bell Labs. The system targets the formal verification of multithreaded software written

Sensor networks

WSNs Head to Himalayas

At press time a team of scientists was heading to the Himalayas to deploy innovative wireless sensor networks (WSNs) in several landslide-prone regions of the world's tallest mountain ranges to provide real-time warnings for often deadly deluges.

The development of these WSNs began about five years ago when an interdisciplinary team of researchers from Amrita University, Kerala, India, combined efforts to design warning systems that would alert people living in landslide-prone areas. The team was made up of computer scientists, Earth scientists, and energy experts and was led by Maneesha Sudheer Ramesh, a founding member of ACM-W India.

The warning system uses WSN technology to issue real-time warnings up to 24 hours prior to an impending landslide, thus facilitating evacuation and disaster management. During the creation of the WSN, Ramesh and her team built a working lab capable of mimicking a landslide. Published papers about the work intrigued the scientific community as well as the university's chancellor, Sri Mata Amritanandamayi Devi, who was the first to proclaim the technology worthy of real-world applications. "Let us actually deploy the wireless sensor network in the field and enable it to save lives."

The first WSN system was deployed in Munnar, Kerala, in June 2009, and has successfully delivered a high level of safety to citizens living in at-risk areas. While the team faced several challenges, including making the system energy sustainable during severe monsoon rains, Ramesh said she was surprised at the "immense amount of psychological safety and security the system seems to have given to the local population, which we never expected since the local population is not that tech savvy."

Last December, the team was awarded the top prize for rural innovation by the Government of India.

—Diane Crawford

in C, the language used for about 96% of all spacecraft software.

"A mission failure often has multiple small triggers that combine in unsuspected ways," he says. "By meticulously fixing the small, relatively benign issues with the same determination as the larger issues, we make sure that serious problems become much less likely."

Those problems are further exacerbated by the organization's now deeply rooted commitment to a centralized computing architecture. Looking ahead, Manning thinks the NASA team will need to rethink its architectural approach for the next generation of robotic flight missions. "Going forward, I would take a more distributed approach," he says.

For now, the team will continue to fine-tune Curiosity's work from a distance over the next several months. But no matter how well the software works, they know full well that space exploration is an inherently unpredictable business—especially on Mars, where wild temperature swings and changes in atmospheric pressure can ruin delicate scientific instruments. To control the onboard temperature, the engineering team developed thermal "catcher's mitts" (as Manning describes them) on the back of the machine, consisting of liquid freon pumped through a closed loop and warmed from the hot plutonium rocks that power the rover.

In order to model as many different scenarios as possible, the team is constantly running so-called soft simulations with dedicated machines analagous to the onboard RAD750 machine. The team also maintains a full replica of the rover on Earth in a testbed environment to troubleshoot problems and rehearse potential future maneuvers. If all else fails, the rover also carries a fully redundant version of its onboard computer, ready to swap in at a moment's notice in case of system failure. "We work hard to make sure the vehicle doesn't come to its knees if it has a small complaint," says Manning.

For all its autonomy, Curiosity still depends heavily on regular communication with its handlers back on Earth. To stay connected with Mission Control—up to 240 million miles, or 21 light minutes away—it relies on a sophisticated interplanetary infrastructure consisting of Earthbound radio transmitters and receivers, and a constellation of satellites orbiting Mars.

For all its autonomy, Curiosity still depends heavily on regular communication with its handlers back on Earth.

NASA put the initial Mars communications system in place with the Mars Global Surveyor mission communicating to the Spirit rover in 2004. Since then, it has made steady improvements to mitigate the ongoing problems of data loss, such as space-link noise, interfering spacecraft, and unpredictable technical problems at the relay spacecraft and ground stations.

Complicating matters further are the limited transmission windows between Mars and Earth due to orbital constraints; usually Curiosity can connect to the network only 2–4 times per sol (astronomer-speak for a Martian day), transmitting an average of 64 megabytes/sol broken into packets, Internet-style.

"We are bandwidth limited," says Sandy Krasner, a NASA software systems engineer who has been working on the Mars project for the past 10 years, "so we have to optimize the use of our downlink as much as possible."

Given the high cost of retransmitting data, the network is designed to focus on error detection and correction, and maximizing loss tolerance. The system sets a maximum file size of one quarter of a megabyte on command files sent to the spacecraft; larger files are broken into smaller datasets and concatenated onboard. To ensure the integrity of data received by the rover, the system also detects and corrects for errors at multiple levels. Data is transmitted in 56-bit blocks assembled into variable-length frames up to 1 kilobyte. The team also tries to tolerate faults in data received from the spacecraft, accepting partial transmissions of image data, for example, where an occasional pixel may get lost in space.

NASA is now working on a more distributed network protocol known as Disruption-Tolerant Networking (DTN) that distributes data across a network of nodes so that any delays or transmission failures can be corrected quickly by retransmitting the data. NASA hopes this architecture will make future interplanetary communication more efficient.

This ongoing network connectivity enabled the programming team to keep tweaking the rover's software well after the mission's launch date on November 26, 2011, sending updates to the onboard computer using a relatively low-tech solution: compressed binary files.

Last June, two months before landing, the team sent up its final in-flight software update while the capsule was hurtling through space at 13,000 miles per hour.

Manning remembers the satisfaction of looking on from a distance of 20-odd light years as rover installed the software and restarted, ready to strike out for parts unknown. "Boot it up and away we go." C

Further Reading

Sky Crane Video;
http://www.youtube.com/watch?v=N9hXqzkH7YA

G.J. Holzmann
The Spin Model Checker—Primer and Reference Manual, Addison-Wesley, Reading MA, 2004, ISBN 0-321-22862-6.

Mars Science Laboratory (NASA site)
http://www.nasa.gov/mission_pages/msl/index.html

G. Groth
Software on Mars, ***Communications of the ACM 55***, 11 (Nov. 2012), 13–15; http://cacm.acm.org/news/156591-software-on-mars/fulltext

B. Cichy
2010 Workshop on Spacecraft Flight Software, California Institute of Technology; http://win-dms-ms1.caltech.edu/five/Viewer/?peid=476727664f1b4d8390d3ab37670ababd

Alex Wright is a writer and information architect based in Brooklyn, NY.