AdaCore CASE STUDY

Raising the Reliability of Scientific Space Exploration

How Ada is helping the Laboratory for Atmospheric and Space Physics minimize risk, reduce cost, and assure reliable control of software used for spacecraft and instrument operations

Customer
The Laboratory for Atmospheric and Space Physics (LASP) at the University of Colorado Boulder (CU Boulder)

Challenge
Upgrading and maintaining a mission-critical command and control system for scientific instruments aboard spacecraft

Solution
The Ada programming language, AdaCore's GNAT Pro Ada development environment, and AdaCore technical support

Results
LASP's highly reliable and maintainable OASIS-CC command and control system has stood the test of time for over 35 years and should continue to do so for decades to come

The Laboratory for Atmospheric and Space Physics (LASP) at the University of Colorado Boulder was founded in 1948, a decade before NASA. It began as the Upper Air Laboratory, using captured German V2 rockets to send scientific instruments into our planet's upper atmosphere.

Today, LASP continues to focus on scientific research, both in the earth's upper atmosphere and beyond. LASP partners with government agencies like NASA and the National Oceanic and Atmospheric Administration (NOAA), commercial entities like Ball Aerospace, and other universities like the University of Arizona and Northern Arizona University, to develop, test, and operate spacecraft and scientific instruments.

Projects in which LASP has participated include NASA's Kepler space telescope mission, the Mars Atmosphere and Volatile Evolution Mission (MAVEN), the Solar Radiation and Climate Experiment (SORCE), the Total and Spectral Solar Irradiance Sensor (TSIS-1) currently aboard the International Space Station (ISS), the Emirates Mars Mission (EMM), and many others.

A highly scalable command and control system

One of LASP's key tools for fulfilling its mission is the Operations and Science Instrument Support–Command & Control (OASIS-CC) software suite. OASIS-CC is LASP's in-house-developed, ground-based, real-time command and control system for spacecraft and instruments. It can be scaled from low-level development through integration, test, and flight ops. LASP has been using OASIS-CC in all its mission roles for more than thirty years.

The OASIS-CC project began in 1985 and chose Ada as the implementation language. Although Ada was relatively new at the time, its software engineering benefits were clear to aerospace systems procurement agencies and developers. The language has been used on OASIS-CC ever since.

According to Jason Gurgel, the OASIS-CC program manager at LASP, OASIS-CC uses a client server model. The core server makes network connections to scientific instruments, spacecraft systems, ground control systems, other ground stations, ground instrumentation, simulators, and other systems. The OASIS-CC client GUIs (Graphical User Interfaces) allow users to design workspaces to display command control panels, telemetry data, and additional information.

OASIS-CC can be used to develop and test scientific instruments in the laboratory and control those same instruments during missions aboard spacecraft. By using OASIS-CC throughout the life of a program, project managers can greatly reduce both risk and cost by leveraging re-use.

Mission-critical software must remain reliable

The heart of the OASIS-CC system—the client server portion—is written in Ada. It has been certified as NASA Class B software (Non-Human Space Rated Software Systems or Large Scale Aeronautics Vehicles). One of LASP's big challenges, according to Gurgel, is making sure that the mission-critical software remains reliable.

LASP is constantly modifying OASIS-CC to control new instruments. And they need to be sure those expensive instruments will work perfectly throughout their mission when doing so. Neither LASP nor their partners can afford to have undefined code behavior or other bugs waste the money spent developing and testing an instrument and sending it into space.

"Naturally, any changes or new features have to be thought about and then thoroughly tested," Gurgel says. "That's just the nature of the software; it's mission-critical."

Keeping a 30-year-old system modern and efficient

One of the big challenges of maintaining the reliability of a 30-year-old real-time system is keeping it up-to-date, manageable, and efficient. That's why, in 2010, LASP decided to undertake a major overhaul of OASIS-CC.

Until then, OASIS-CC had been a monolithic system—a single program written in Ada 83. To make the system more efficient and maintainable, LASP decided to repartition it and update its code to a more modern version of Ada.

For the latter, LASP needed a new compiler. They had been using a Verdix/Vads Ada 83 compiler from the 1980s. What they wanted was a modern Ada compiler supported by a vendor with deep Ada expertise. They were looking for a partner who would aid them in their Ada upgrade and help them make sure their software remained certified to NASA Class B.

A total system migration and upgrade

LASP decided to refactor their entire code base and migrate it from a SPARC/Solaris system to a Linux platform. They removed all the GUI portions from the Ada code and reworked the rest as a distributed client server, upgrading the code from Ada 83 to Ada 95. Then, they wrote brand new C++ code for the front end and the displays on all the GUIs.

For their new Ada compiler, they chose AdaCore's GNAT Pro compiler.

"The main reason was support," said Gurgel of why they went with GNAT. "We wanted a supported compiler. We also wanted Ada language support to help with the transition [to Ada 95] and to check the box for keeping that software certification in place."

"The main reason [we went with AdaCore's GNAT Pro] was support. We wanted a supported compiler. We also wanted Ada language support to help with the transition and to check the box for keeping that software certification in place."

— Jason Gurgel, OASIS-CC program manager, LASP

A full user spectrum of customer support—direct from AdaCore's expert developers

AdaCore realizes that teams exist on a spectrum, from those who simply wait for new versions and features to be released through AdaCore's continuous release download page, to those who require ultra-reliable, long-term support.

For the latter, AdaCore provides GNAT Pro Assurance, a version of the standard GNAT Pro Enterprise product augmented by a service known as "sustained branch"—a stable development line in which critical fixes can be back-ported and accompanied by a thorough impact analysis. GNAT Pro Enterprise customers, like LASP, are welcome and encouraged to make technology enhancement requests. Through this system, many customer suggestions have been implemented and incorporated into GNAT Pro over the past several years.

For all its product subscriptions, AdaCore provides industry-leading support directly from the experts who develop AdaCore's GNAT Pro technology. Not only do customers receive timely, expert support, this process also helps AdaCore's developers better understand how the technology is being used. And that, in turn, helps them provide both pragmatic guidance and useful technology upgrades.

Gurgel calls AdaCore support the biggest benefit of porting to the GNAT compiler. "It was a huge benefit during the upgrade, when we really needed expert support," he said. "Now, things are pretty stable. We′ve had a few issues come up, and AdaCore has always been responsive. We use the support much less now, but it′s good to know AdaCore is there when we need them."

The Ada language—designed for high reliability in mission-critical, real-time systems

Gurgel says Ada's features make it the best language choice for the mission-critical portions of OASIS-CC. Two features he cited as essential are concurrency (tasking) and data type security.

Concurrency support

A concurrent program is one in which different parts of the program can execute in interleaved fashion on a single processor/core, or with actual parallelism on a multiprocessor/multicore architecture. Advantages include increased program throughput and better input/output responsiveness, and, for a real-time program, a software design that aligns to the external environment with which the program is dealing. The challenge is that concurrent programming is beset with issues (such as deadlock, race conditions, livelock, and corruption of shared data) that do not arise with sequential programming. Ada's concurrency support gives the developer the expressiveness to address this challenge and to write reliable, portable, predictable, and efficient real-time software.

In Ada, the unit of concurrency is known as a task. A task is a module that has a specification (an interface visible to other program units) and a separate body that encapsulates its local data and behavior. A task executes as a separate thread of control with its own stack. Its interface consists of its ″entries″ (names for synchronization points), which can be called from other tasks. A task reaching a synchronization point is suspended (the so-called ″rendezvous″) until another task takes some action to unblock it.

Ada features that make the language ideal for developing real-time multi-tasking programs include:

- A powerful but easy to use protected object abstraction, including an object locking policy that maximizes schedulability for a set of periodic and event-driven tasks; and

- An efficient subset of the concurrency features (the so-called Ravenscar profile) that facilitates formal demonstration of program properties and that is appropriate for small-footprint embedded target platforms.

Data type security

Ada's data type facility helps users detect data misuses early, with a combination of compile-time checks to make sure that a data object is only accessed via operations permitted for its type, and run-time checks to enforce type-specific constraints.

For example, in a weakly typed language an integer can be treated as a pointer, and vice versa, leading to errors that can be time-consuming to correct, and, if undetected, can introduce security vulnerabilities. Ada prevents these errors at compile time. In systems-level programming where such low-level data manipulation may be necessary, Ada provides the relevant features, but with explicit syntax to make the programmer's intent clear.

A data type may be considered a compile-time invariant property of a data object. Additionally, a data object has, either implicitly based on its type or else explicitly provided at its creation, a run-time constraint that establishes the permitted set of values that the object can take. Through run-time checks, Ada ensures that each data object only takes values that satisfy its constraint. This prevents insecurities such as buffer overrun, integer overflow, and scalar range violations.

Ada training for new employees

Another challenge for LASP is one that many Ada organizations face when hiring new employees. "Few candidates know anything about the language," says Gurgel, "but if we want them to work on the system, we've got to train them in Ada. Our OASIS-CC team is small, so that's a big challenge for us."

Help with Ada training is also a big benefit of partnering with AdaCore. AdaCore provides live training, as well as online self-study courses through their learn.AdaCore.com website.

AdaCore also offers a broad library of how-to books in PDF format to help software engineers become proficient with Ada. These include publications tailored for C, C++, Misra C, and Java developers, and several which address specific industries and industry standards. All can be downloaded from AdaCore's website at adacore.com/books.

LASP has availed itself of several of AdaCore's training opportunities, including a live, 5-day Ada Fundamentals course. The feedback that LASP provided to AdaCore regarding the course was overwhelmingly positive. Participants found the exercises helpful and had no trouble understanding the concepts presented.

"We don't have a formal training program," says Gurgel. "From my experience in hiring a few new people, I think it's a little difficult for them to pick up Ada without some kind of basic training. If you just throw them in the deep end, I think they'll be overwhelmed at first. But once they get a little AdaCore training, it's fairly easy for new team members to adapt to Ada."

"Once they get a little AdaCore training, it's fairly easy for new team members to adapt to Ada."

— Jason Gurgel, OASIS-CC program manager, LASP

Outstanding system reliability and cost-effectiveness

According to Gurgel, OASIS-CC's robust baseline has stood the test of time. "We make sure we maintain that baseline moving forward," he says. "The software's been around for so long. It's been a very reliable system."

It has also been extremely cost-effective. "If we were to go out and get a commercial command and control system, that would cost us a lot of money, a lot of time configuring the system, and a lot of time training," says Gurgel. He maintains that the OASIS-CC and Ada fulfill all their needs for reliability and maintainability. "Ada does everything we need it to do. Plus, it lets us do everything we need to do ourselves. As far as the server backend goes, with its tasking features, its reliability, and its type safety, I think Ada is still the best and only choice for this critical software."

"With its tasking features, its reliability, and its type safety, I think Ada is still the best and only choice for this critical software."

— Jason Gurgel, OASIS-CC program manager, LASP

Years of reliable service still to come

Gurgel says the time and effort LASP invested in rearchitecting and upgrading the system ten years ago is still paying dividends today. He expects it will continue to do so for many years.

"That effort has extended the system's life well into the future," he says. "We don't have any plans to rewrite it. It's definitely fulfilling our needs."

AdaCore has a long history of providing tools and expertise to the Space industry.

For more information please visit: adacore.com/industries/space.

And check out our latest book on AdaCore Technologies for Space Systems Software.

AdaCore Technologies for
SPACE SYSTEMS SOFTWARE v1.0
Supporting qualification for ECSS-E-ST-40C and ECSS-Q-ST-80C
Benjamin M. Brosgol & Jean-Paul Blanquart