AdaCore CASE STUDY

How Masten Space Systems is Using Ada and SPARK to Land on the Moon's South Pole

When Masten Space Systems was awarded a NASA contract to land scientific payloads on the Moon, they knew they would need software that could be proven reliable —and proven quickly.

All the mission-critical flight control software for their XL-1 Lunar Lander had to work perfectly. And to top it all off, Masten was working with an extremely constrained schedule and budget.

That's why they chose Ada and SPARK.

Customers

Masten Space Systems - a developer of reusable vertical-takeoff/vertical-landing (VTVL) rockets

Problem

Masten's XL-1 Lunar Lander will transport a suite of scientific research payloads to the lunar south pole. It boasts numerous small embedded computers called electronic control units (ECUs) that turn the power on and off to various components, control the engines and thrusters, and interface with sensors. Developing the software for these small, heavily constrained ECUs presents a number of challenges. The software will run in a "bare metal" configuration, and it has to be extremely reliable and easy to maintain. The code also needs to be modular and reusable, to avoid duplication of effort and to reduce development costs and timeline.

Solution

To develop their mission-critical flight control software, Masten chose the Ada and SPARK programming languages, together with AdaCore's GNAT Pro integrated development environment and the SPARK Pro static analysis tool suite.

Results

While the XL-1 Lunar Lander project is still in development, Masten already sees substantial value in the capabilities that Ada, SPARK, and AdaCore's products and support have brought to their embedded project. They expect to realize a reduction of at least 20 to 30 percent in verification and validation time and in overall development costs and timelines, as well as a 20 to 30 percent increase in code reuse.

The challenges of landing on the lunar south pole

In April of 2020, NASA awarded Masten a contract to transport a suite of scientific research payloads to the lunar south pole. The NASA Commercial Lunar Payload Services (CLPS) project is a $75.9 million contract that covers everything from the lab to the surface of the Moon, along with a host of payload services after landing.

Geologically, the lunar south pole is highly active and an area of great interest to scientists. The terrain is very uneven, making it a difficult place to land a spacecraft and explore with a rover. Once in lunar orbit, the XL-1 must be able to navigate to the polar region, determine where to land, and land safely, avoiding the numerous hazards in the area. It must touch down within a very narrow corridor—where the scientific interest lies, not five kilometers away. And it must do all of this autonomously.

Developing mission-critical spacecraft software to a tight schedule

Masten's lander boasts a large collection of computers, including numerous smaller computers called electronic control units (ECUs).

The ECUs are embedded systems that interface with the hardware on the spacecraft. They turn the power on and off to various components, control the engines and thrusters, and interface with the sensors.

Developing software for these small, heavily constrained ECUs presents a number of challenges, according to Abhimanyu Ghosh, the avionic software engineer who is leading Masten's ECU software development effort for the XL-1.

First, memory and power are scarce resources on the XL-1, as they are on most spacecraft. To minimize resource consumption, the ECU software will run in a "bare metal" configuration, that is, without an operating system or other facilities that support the application code.

Second, the software must be easy to maintain. Therefore, it must be easy to understand for both technical and non-technical team members.

Third, the software must be extremely reliable in flight. It must be verified to be correct—free from bugs that might threaten the mission and result in the loss of millions of dollars of scientific equipment.

Finally, the software development process has to meet the program's tight project schedule.

"A lot of things that might be acceptable in traditional spacecraft development timelines don't align well with our program goals," says Ghosh. To meet their schedule, Masten needed their code to be modular and reusable, so they could avoid duplication of effort.

Therefore, it was crucial to choose the right programming language and the right set of development tools that would support a bare metal configuration, enforce correctness of the code by design, facilitate rapid verification and validation, and maximize code reuse.

Choosing the right language (Why C/C++ was not the answer)

"The problem with using C or C++ for safety-critical and mission-critical applications is that these languages are fundamentally memory-unsafe and non-concurrency-aware."

— Abhimanyu Ghosh, Avionic Software Engineer, Masten Space Systems

According to Ghosh, Masten's choice of programming language for their embedded ECUs came down to the amount of effort that would be required.

Ghosh said they could have used C and C++. There are, after all, many spacecraft flying with C/C++ flight software. They are even used for some applications on the XL-1. "The problem with using C or C++ for safety-critical and mission-critical applications," he said, "is that these languages are fundamentally memory-unsafe and non-concurrency-aware."

Organizations who use C/C++ for such applications typically require the use of an extensive coding standard for the language. Examples include MISRA C, MISRA C++, the Lockheed JSF AV C++ Coding Standards, and the JPL Institutional Coding Standard for the C Programming Language.

"Use of such a coding standard," says Ghosh, "would significantly drive up the time and cost of verification and validation for Masten's ECUs."

Instead of C or C++, Masten chose Ada and SPARK, together with GNAT Pro from AdaCore.

SPARK is a language plus a toolset. The SPARK language is a formally analyzable subset of Ada. Its primary design goal is to provide the foundation for sound formal verification and static analysis. It embodies a large subset of Ada 2012 while prohibiting certain features that are common sources of defects and not amenable to static verification. The SPARK toolset brings mathematics-based confidence to software verification through use of formal methods.

GNAT Pro is AdaCore's complete solution for producing critical software systems where reliability, efficiency, and maintainability are essential. It provides an integrated development environment for Ada and offers a suite of tools and libraries for developing large, mission-critical applications. There's also optional support for C and C++.

Why Masten chose AdaCore

Masten decided to put SPARK to use where it would have maximum value: in code that is critical to the mission, code that must be flawless.

The XL-1 is not a totally SPARK-driven environment. Ghosh and his colleagues are trying to strike a balance between (1) making use of existing codebases that were not written in Ada or SPARK, and (2) using the features of SPARK where they need to verify that the code is absolutely correct.

Ghosh lists several reasons why Masten chose AdaCore's tools.

One reason is AdaCore's business model, which is based on open-source software plus support. "That's definitely something I feel is a good thing," Ghosh says. "It's beneficial to the company, and it's also beneficial to the community, and to the customer."

A second reason is that AdaCore support comes directly from the developers of the technology, as opposed to the traditional "first line" support. "Having access to the folks who actually designed the tools—who really understand the detailed technical jargon—is extremely useful. It's going to be even more useful going forward, as we're only at the beginning of ramping up on AdaCore's technology."

Third, Ghosh was impressed with the modularity of the AdaCore toolchains. "A lot of things can be adjusted to the customer's needs," he says. "It's not a monolithic solution like so many others in the embedded market that are aimed at a narrow use case. Most vendors say, 'Here, take this, and this is how you're expected to develop.' But as soon as you try to do advanced things with the tools, you run into a brick wall. That's not the case with GNAT Pro and SPARK."

Ghosh also praised the high level of AdaCore's documentation, and he greatly appreciates the fact that all the source code is provided.

Ramping up with SPARK Pro and GNAT Pro

Masten has made good initial progress in defining their software architecture. They are currently refining their full system-level and subsystem-level requirements and flowing those down into individual component-level software requirements for the XL-1.

Ghosh says they're trying to fully understand what they're building before they start coding. As part of that effort, Masten has been using SPARK and GNAT Pro for R&D and prototyping, alongside C and C++.

"We've been able to run SPARK in a bare metal environment using the Ravenscar profile," says Ghosh. He also says that they've implemented some tasking and a top-level executive for controlling general-purpose IO and serial ports, and they've been verifying core behavior and the processor interface.

Ghosh says they're trying to fully understand what they're building before they start coding. As part of that effort, Masten has been using SPARK and GNAT Pro for R&D and prototyping, alongside C and C++.

One thing that struck Ghosh during this early development is how AdaCore's tools stand out in the embedded world.

"Many embedded tools are very rigid," he says. "The developer is locked into a graphical environment where you click a certain button or do specific things to make certain things happen. They're built to be easy for the developer to use during initial development stages. Where they falter is when they're used in continuous integration or cloud-based builds."

"SPARK Pro and GNAT Pro are far more flexible. You can use the IDE (GNAT Studio), but it's by no means required. Everything you do within the IDE can be scripted in some way. The pieces of the AdaCore tool stack are very modular. That has major benefits not just for development, but for integration as well."

Ghosh also likes AdaCore's use of standard open-source software development techniques, such as using Makefiles.

Improvements of at least 20 to 30 percent

While it's too early for Masten to quote quantifiable results, Ghosh can already see substantial value in the capabilities that Ada and SPARK bring to an embedded project.

"We're probably looking at a reduction of at least 20 to 30 percent, if not more, in verification and validation time and in overall development costs and timelines… I expect a 20 to 30 percent increase in code reuse as well."

— Abhimanyu Ghosh, Avionic Software Engineer, Masten Space Systems

"We're probably looking at a reduction of at least 20 to 30 percent, if not more, in verification and validation time and in overall development costs and timelines," he says. "And that's compared to very well written, very well optimized C++ code, which is very hard to come by these days. I expect a 20 to 30 percent increase in code reuse as well."

Ghosh also believes SPARK will help Masten simplify their code. They are borrowing from functional safety principles to come up with an architecture that gains them value while reducing complexity.

"That's not captured in the software effort in many programs right now," he says. "If you look at the JSF C++ document, it's a minefield. That's very prevalent in the C/C++ world. People write these elaborate 100-page-long company coding standards. With Ada and SPARK, you can use automation to enforce correctness at the software level and not put people through this."

Better traceability

One of the things Ghosh appreciates in Ada and SPARK is the clarity of the syntax. He believes it makes for better traceability between requirements and code.

"A lot of requirement details get lost in the implementation of a C/C++ program," he says. "You certainly can't take a C++ program and show it to a systems engineer and expect them to understand what it's doing. There's a much better sense of traceability between requirements written in English by a systems engineer and the things you see in the source code listing when you write in Ada. With Ada's extremely modular nature, you're specifying in a very 'systems engineering-oriented' manner. The inputs, outputs, and contracts are inherent in a module, which you can't really do with C++ in a meaningful way."

There's a much better sense of traceability between requirements written in English by a systems engineer and the things you see in the source code listing when you write in Ada.

— Abhimanyu Ghosh, Avionic Software Engineer, Masten Space Systems

Ghosh says Masten's expectations for Ada and SPARK have definitely been met and even exceeded, especially in terms of the support they've received and the ease with which they've been able to ramp up on the technology.

Big milestones on the horizon

One big milestone Masten wants to pass very soon is the implementation of error detection and correction (EDAC) features on their processor. They'll also be implementing a network stack and an interface to the network stack from within the Ada application.

After that, another important milestone will be integrating new Ada-based code into their existing C/C++ software environment. They'll take actual Ada/SPARK user applications and inject them into the C/C++ codebase.

“There's a significant amount of development ahead,” says Ghosh, “but I'm definitely looking forward to it.”

Contact **info@adacore.com** if you would like to speak to a space expert about AdaCore technologies for space systems

About Masten Space Systems

Born in California's Mojave Desert in 2004, Masten Space Systems develops technology to accelerate space ecosystems on the Moon, Mars, and beyond. Their goal is to unlock the value in space to ultimately benefit humans on Earth. Masten's reusable rockets, lunar landers, and supporting technologies break down the barriers to space for government, defense, and commercial customers.

Masten began prototyping its vertical-takeoff/vertical-landing (VTVL) rockets long before SpaceX and Blue Origin made that concept well-known. Thanks to their pioneering work in VTVL, they won the NASA Centennial Northrop Grumman Lunar Lander X-Prize Challenge in 2009.

Since then, Masten has worked with NASA and other customers to advance Technical Readiness Level (TRL) for a wide variety of space technologies. Customers can put their payload on one of Masten's rockets and test it on Earth. Masten provides a unique service that allows their customers to control Masten's rocket through the payload itself.

One payload was the Jet Propulsion Laboratory's Lander Vision System (LVS), which was recently used to land the Perseverance rover on Mars. The LVS was tested and its TRL advanced through flights on Masten's Xombie rocket.

In 2020, Masten was awarded a NASA CLPS contract to land scientific research payloads at the South Pole of the Moon. That landing is scheduled to take place in late 2023.

About AdaCore

Founded in 1994, AdaCore supplies software development and verification tools for mission-critical, safety-critical, and security-critical systems.

Over the years, customers have used AdaCore products to field and maintain a wide range of critical applications in domains such as commercial and military avionics, automotive, railway, space, defense systems, air traffic management/control, medical devices, and financial services. AdaCore has an extensive and growing worldwide customer base; see www.adacore.com/industries for further information.

AdaCore products are open source and come with expert online support provided by the developers themselves. The company has North American headquarters in New York and European headquarters in Paris. www.adacore.com.