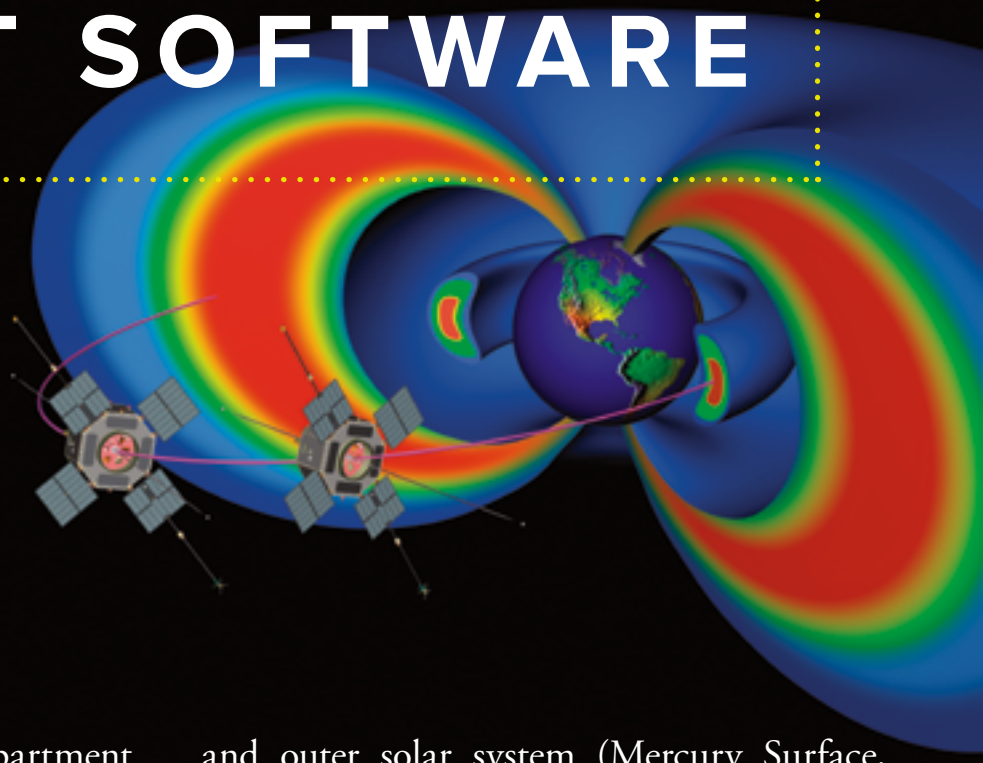
THE ROAD TO THE NEW FLIGHT SOFTWARE

BY CHRISTOPHER KRUPIARZ



In 2004, my group in the Space Department of the Johns Hopkins University Applied Physics Laboratory (JHU/APL) was presented with a critical opportunity and challenge. We had successfully developed and deployed spacecraft flight software on a number of NASA missions over the previous decade. They included the Advanced Composition Explorer, a spacecraft at a point of Earth–sun gravitational equilibrium almost a million miles from Earth; an Earth orbiter (Thermosphere, Ionosphere, Mesosphere Energetics, and Dynamics mission); the Near-Earth Asteroid Rendezvous spacecraft; the twin Solar Terrestrial Relations Observatory probes; and missions destined for the inner and outer solar system (Mercury Surface, Space Environment, Geochemistry, and Ranging spacecraft and New Horizons). Over the course of that decade, our flight software had become tightly coupled, with changes in one application affecting others. We were able to reuse the software during this time, but doing so depended on reusing the same avionics and the same personnel. When teams or hardware changed, the software was difficult to apply to new missions without substantial modification. It was clearly time to revamp our core architecture, but we wanted to do it in a way that preserved many of our existing applications while modernizing the overall structure.

Image Credit: NASA

Cutaway model of the Earth's Van Allen radiation belts with the two satellites from NASA's Van Allen Probes, the first mission to fly the new Applied Physics Laboratory flight-software architecture.

Photo Credit: NASA Marshall Space Flight Center/David Higginbotham

The Robotic Lunar Lander fires its onboard thrusters to carry it to a controlled landing using a pre-programmed descent profile. Guidance and control (G&C) development at the Applied Physics Laboratory is a collaborative effort between the flight-software group and the Marshall Space Flight Center G&C analysts' group.

Bruce Savadkin, my group supervisor at the time, recognized this need. Through proposals to the JHU/APL Internal Research and Development board, he successfully acquired funds to work toward this goal. Our first step was to identify software that would decouple our software architecture and allow individual applications to operate independently. This study led us to select an architecture developed by NASA Goddard Space Flight Center, called the Core Flight Executive (cFE).

Integrating the Core Flight Executive

CFE is a suite of software that provides multiple services to flight-software applications. A key to these services is a software communication bus, or transfer interface, that makes a modular, decoupled architecture possible. Instead of individual applications calling functions within other applications and creating intractable dependencies among them, cFE communication occurs via message passing. An application publishes messages and subscribes to messages on a software bus, providing a single input source to the application. With a well-defined message dictionary that various applications can understand, this provides a straightforward way to plug and play new applications into a system.

Once that middleware was selected, we began adapting our flight software to the cFE concept. Transitioning to a new architecture took a significant amount of rethinking. We had been working with our current architecture for years; we fully understood its idiosyncrasies, its advantages, and its limitations. The new architecture required a new way of thinking. Additionally, some in the group were reluctant to change. Their reluctance came with strong arguments, including, "We just launched a probe to Mercury. Why change a successful architecture?" and "Why not wipe the slate clean and rebuild from the bottom up?" (The answer to the first question was, "We need to improve our ability to reuse code to lower costs," and to the second, "Too expensive.") So our development process was not only technical. It included a necessary series of discussions to bring those who were reluctant to change onboard.

As the initial lead on the project, it was my responsibility to handle these questions and to find a way forward for the design. Leading a team on a research effort this large was a new experience for me. Unfortunately, I quickly learned lessons on how not to do it. Whereas my previous efforts with large teams had specific requirements and goals, this research effort was much more open ended; we had to answer the question, "What is good?" before we could build the software. So my usual project management method of trying to reach an agreement on small issues while we all agreed on the larger purpose immediately ran into trouble. Not surprisingly in hindsight, when you ask a group of experienced flight-software engineers what a good architecture is, you get multiple answers. As a result, we had many false starts that resulted in slower progress than I had originally hoped.

To address the problem, we identified a couple of key personnel who had strong technical reputations within the group as well as extensive flight-software experience and asked them to define a path forward. While it would not meet the impossible goal of unanimous consent, we knew that their experience and the trust they inspired meant it would be well received. At the end of the effort, the team had encapsulated enough of our heritage code in cFE applications to demonstrate that we could have the best of both worlds: a modular architecture that leveraged our past success. We had shown that cFE was adaptable to our architecture. Now we just needed a mission to prove it.

AT THE END OF THE EFFORT, THE TEAM HAD ENCAPSULATED ENOUGH OF OUR HERITAGE CODE IN CFE APPLICATIONS TO DEMONSTRATE THAT WE COULD HAVE THE BEST OF BOTH WORLDS: A MODULAR ARCHITECTURE THAT LEVERAGED OUR PAST SUCCESS. WE HAD SHOWN THAT CFE WAS ADAPTABLE TO OUR ARCHITECTURE. NOW WE JUST NEEDED A MISSION TO PROVE IT.

As it happened, we had two: the Radiation Belt Storm Probes (now called the Van Allen Probes) and the Robotic Lunar Lander program.

The Van Allen Probes

The Van Allen Probes are twin spacecraft studying the Van Allen radiation belts. They are also the first mission to fly the new JHU/APL flight-software architecture based on cFE. Mark Reid, flight-software lead for the mission, was instrumental in advancing the architecture. He began his prototyping work in Phase A, working closely with the mission operations and integration and test teams—the ones who interact most with our software.

Naturally, they were accustomed to operating a spacecraft in a certain way. Familiarity with institutional procedures from mission to mission is a key to the success of our spacecraft. When introducing cFE, Mark focused on ensuring it would not disrupt those procedures. He avoided cFE features that fit Goddard's operational model but would have been too disruptive of APL's procedures. Mark also did early benchmark testing of cFE operations to understand their impact on resource utilization. We expected to see an increase in processor and memory usage, since we understood that cFE is more complex than directly coupling software. Mark's team's measurements showed that cFE would work within the computing constraints of the spacecraft. He also found that focusing on software that was not dependent upon external communication made it possible to reuse a significant amount of our code base while transitioning to the new architecture.

On the whole, the cFE integration was a success. The primary difficulties the team encountered were not with the code itself. Auxiliary tasks that come with managing a large body of code—for instance, version control, bug fixes, and updates—caused the greatest difficulties. Because Goddard was developing its own spacecraft while also supporting cFE, it was understandably difficult for them to respond to requests from outside the organization. Fortunately, Mark and his team developed strong personal relationships with Goddard personnel, which ensured focused responses to our needs.

The Robotic Lunar Lander

The Robotic Lunar Lander development article is a joint effort between JHU/APL and the Marshall Space Flight Center. To understand how cFE operated within the lander, I exchanged e-mails with Gail Oxton, who was the flight-software lead through a significant part of development. She and her team were responsible for developing the guidance and control algorithms that would fly on the test vehicles. Marshall developed the command and data-handling functionality and sensor interfaces.

Guidance and control (G&C) development at APL is a collaborative effort between the flight-software group and the G&C analysts' group. The analysts develop G&C models via MATLAB to accommodate the constraints and requirements of a given mission. Once that work is complete, they turn to Simulink to auto-generate flight code that is then delivered to the flight-software group and integrated into our flight software. For the robotic lander, Gail devised an initial plan to deliver the auto-generated C code for the G&C models directly to Marshall. But as Gail remarked, "That can be a challenge when the G&C analysts and the software team are on different floors, let alone in different states."

She decided instead to implement G&C as an entire cFE application so the interface between Marshall and APL would be solely over the software bus. Needing to define only a small set of messages for communication removed potential

> NOT SURPRISINGLY IN HINDSIGHT, WHEN YOU ASK A GROUP OF EXPERIENCED FLIGHT-SOFTWARE ENGINEERS WHAT A GOOD ARCHITECTURE IS, YOU GET MULTIPLE ANSWERS.

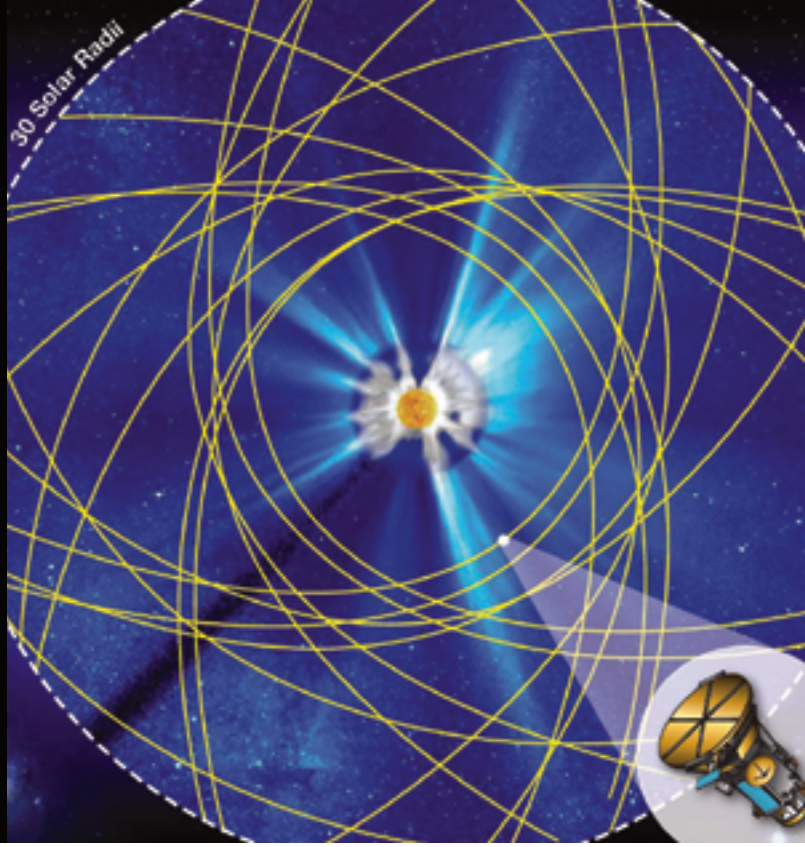


Image Credit: NASA

A simulated view of the sun illustrating the trajectory of Solar Probe Plus during its multiple near-sun passes. The Applied Physics Laboratory is flight-software lead for the project, working to further the Core Flight Executive architecture.

dependencies within the code. This was a step forward in both collaboration and software reusability; it would be the first time we delivered a cFE application externally.

To achieve this solution, Gail developed an interface control document that defined all software bus traffic. This involved a range of data including clock ticks, sensor input, and commands from Marshall's command and data handling to APL's guidance and control, and thruster fire commands, attitude data, and other telemetry from G&C to command and data handling. Over the next few months, each team separately developed and tested their applications. When the APL G&C application was delivered to Marshall, the Marshall team successfully integrated the G&C application in literally a few hours. Gail had one brief, over-the-phone debug session to identify an array indexing problem on day two, but after that the software worked flawlessly. Over time, Gail's team delivered algorithm improvements to Marshall. Each delivery was similarly smooth. The Robotic Lunar Lander continues to have many successful test flights.

When asked about the experience, Gail summed it up this way: "We had no prayer of getting this to work in the timeframe and funding we had without cFE."

Solar Probe Plus and the Future of cFE

As flight-software lead for the Solar Probe Plus project, I am working with my team to further the architecture. We are striving to make the software even more reusable and cost effective through configuration values, parameters, and tables that can reduce the amount of rework from mission to mission, relying instead on configuration variables to modify the software. We are also working with the Van Allen team to avoid some procedural difficulties encountered on that project.

CFE and Goddard's larger Core Flight System, of which cFE is a part, continue to achieve recognition outside Goddard. It is not only performing flawlessly on the Van Allen Probes and the Marshall lander, but it is also being used on projects such as Johnson Space Center's Morpheus effort, the Ames Research Center's Lunar Atmosphere and Dust Environment Explorer, and Goddard's Lunar Reconnaissance Orbiter, Global Precipitation Measurement spacecraft, and the Magnetospheric Multiscale mission. CFE can work for an organization that has no existing flight-software experience or architecture; it can also work, as we showed, for an organization with an existing architecture. CFE and the Core Flight System have the potential to serve as a basis for other NASA missions, reducing costs and simplifying the process of developing software for the full fleet of NASA spacecraft. Currently, Goddard has to turn to individual missions to improve cFE on a mission-by-mission basis. What the Van Allen experience has shown us is that Goddard (and NASA in general) has a strong product available for use by the NASA community. As the user base grows, we hope institutional support will grow with it. ●

CHRISTOPHER KRUPIARZ is a member of the Johns Hopkins University Applied Physics Laboratory (JHU/APL) principal professional staff. He is currently the assistant group supervisor of the JHU/APL Embedded Applications Group and flight-software lead for the Solar Probe Plus mission.