Performance
Seeing Things *Differently*

# JETS

## VIRTUALIZATION FOR EMBEDDED SOFTWARE

# VIRTUALIZATION WITH JETS:
## A KEY ENABLER OF DEVOPS TRANSFORMATION FOR AEROSPACE AND DEFENSE

Like all companies in the technology industry, aerospace and defense suppliers are facing increased pressure to reduce cost and increase productivity, while meeting their market's demand for higher overall quality and rapid adoption of new platforms, development paradigms and user experiences.

In recent years, the rapid pace of innovation for consumer tech, together with a new generation of 'digital native' users that demand more capability than previous generations, has left many in the industry playing constant catch-up.

**1**

The commercial software world is shifting again, with over **74%** of companies expected to embrace **DevOps** in the next several years.

https://devops.com/state-devops-adoption-trends-2017/

Seasoned aerospace organizations have already weathered the change from bespoke systems and proprietary languages to COTS and open-source, as well as paradigm shifts from high-overhead waterfall and spiral development models to lightweight and agile methodologies and are considering moving to scalable agile methods. As with previous seismic shifts in the industry, most defense and aerospace firms will have to follow, especially as best-of-breed tools, talent and processes are all refactored to take advantage of **DevOps'** unique productivity and quality improvements. **DevOps** is a collective term for a range of modern development practices that combines loosely-coupled architectures and process automation with changes to the structure of development, IT and product teams. Adoption of DevOps (See Figure 1).

Adoption of **DevOps** includes a shift from long, structured release cycles to continuous delivery, and relies on heavy use of automation to improve productivity and quality. This can be complicated by several factors common to the defense and aerospace industry.

For starters, unlike the fast-moving software-as-a-service (SaaS) and mobile application segments, most aerospace software is developed in a regulated environment, with required processes and practices, standards compliance, formal certification procedures, and occasionally mandated use of specific technologies. Implementing continuous delivery, for example, can be hampered by the need to undergo re-certification prior to upgrades, or when system-of-systems testing is required for traditional 'block' releases.

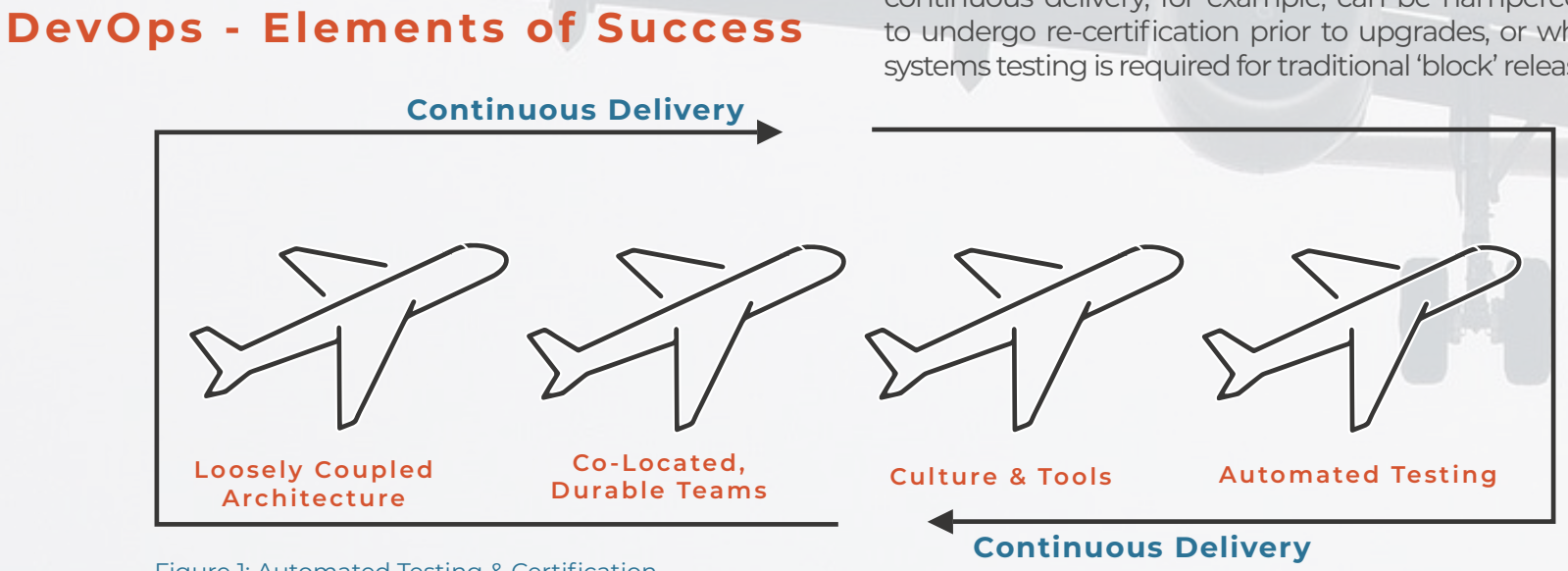


Figure 1: Automated Testing & Certification

Some procurement and contracting models may also mandate specific release strategies, aligned with earlier formal process methodologies like the Capability Maturity Model for Integration (CMMI). Suppliers and the government have invested heavily in these processes, and in many cases the culture and even the toolchains depend on them.

The net result is the critical need to balance regulatory requirements with new automated approaches that meet the spirit and letter of the regulations. In addition to enabling **DevOps** with our customers, Performance is also working with regulatory bodies including the FAA to modify the regulatory process accordingly.

## DISCUSSING COMPONENTS IMPORTANT TO DEVOPS PROCESS

**Aerospace systems also differ from many other industries due to the complex integration of embedded and analog components developed using different processors, operating systems and programming languages.**

Full support for **DevOps'** philosophy of 'automating everything' is complicated by the need to provide the appropriate I/O simulations for architectures that are many times not loosely coupled. Aerospace systems often include legacy components that still provide critical functionality, but were developed many years ago using tools and methods very different from those of today. Programming languages and operating systems that have long dropped out of use in commercial software linger in systems that are not cost-effective to redevelop or discard.

Despite these challenges, there is still a strong rationale for aerospace and defense companies to adopt **DevOps** and a preferred development strategy for both new and legacy technology programs. For starters, **DevOps** can often reduce the complexity of development in ways that significantly contribute to quality and cost savings. In a traditional software lifecycle, feature development and operational tasks like deployment, sustainment and maintenance are loosely coupled. While this model of software deployment gives users tons of new functionality all at once, in practice it has often led to long, expensive development cycles and buggy releases that require lots of subsequent patching.

With **DevOps,** teams focus on making ongoing, incremental and well-understood changes to a living system. Each update, whether a bug fix or new functionality, is integrated into a fully tested, up-and-running system. What's more, the same small engineering team is expected to be responsible for the full lifecycle of a feature or component, from design and development to test, deployment, and maintenance. This type of automated software development enables continuous delivery (See Figure 2).

When something goes wrong, whether in system test or in the field, the team can troubleshoot and fix it right away.

**2** One study showed that adopting a **DevOps**-style development strategy for a major aerospace subsystem was able to reduce the time needed to deploy a minor patch from weeks and tens of thousands of dollars to a few days and a few thousand dollars.

These kinds of productivity and quality gains are possible because **DevOps** teams make heavy use of automation and virtualization to take the place of labor intensive manual processes, hardware resource constraints and hand-offs to independent teams. DevOps organizations often maintain a mandate to 'automate everything,' meaning tasks that can be repeatably performed by software are scripted or programmed into the system's implementation. This includes compilation, system setup and configuration, software deployment, API and unit test, system integration, code coverage, certification and automated reporting of failures.

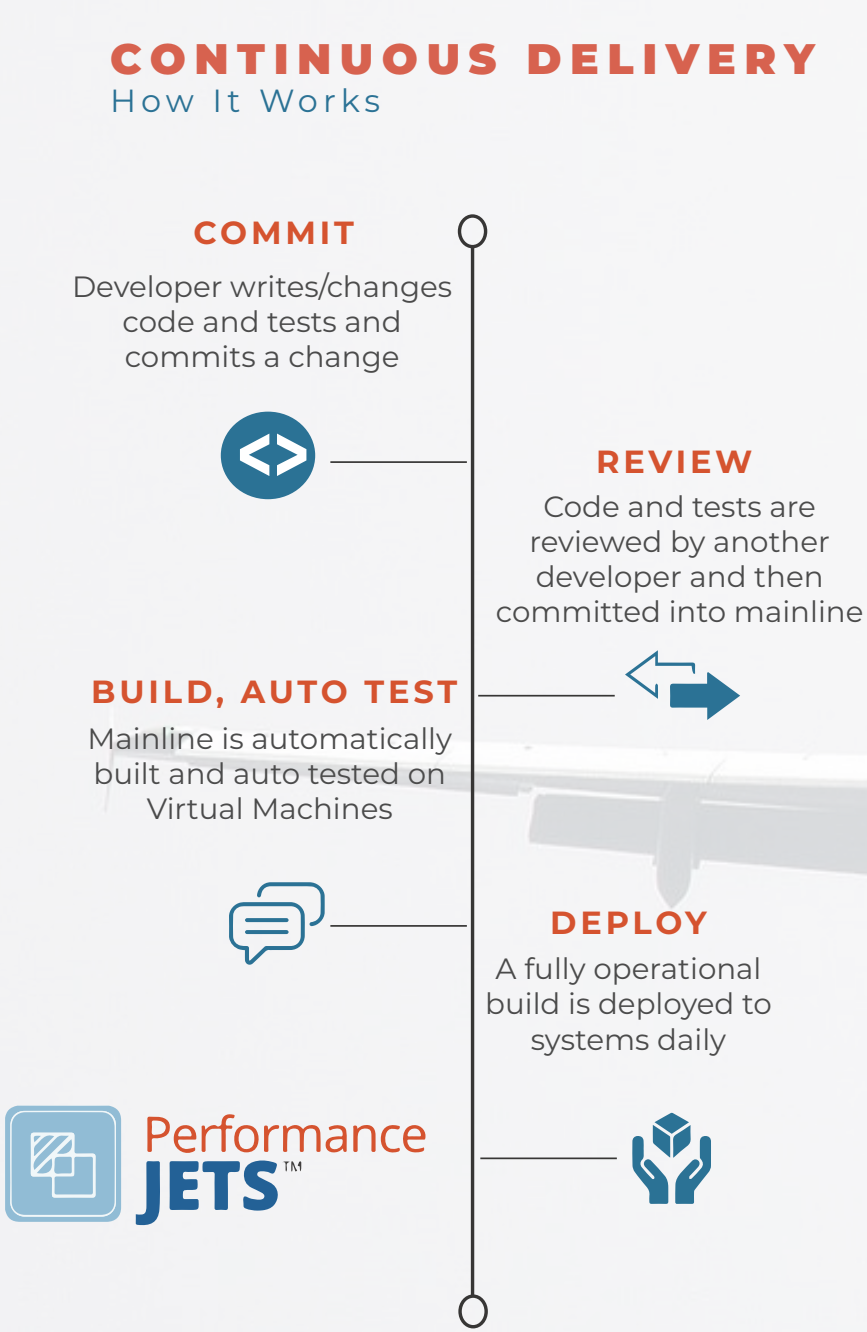


Figure 2: Performance JETS Makes DevOps Productivity and Quality Possible for Aerospace

While it may seem labor-intensive up front, defense and aerospace companies stand to benefit even more from **DevOps** automation than IT and SaaS organizations.

That's because aerospace and similar systems have a much longer lifespan than a typical consumer website or mobile app. While the average lifespan of an iPhone or Android application is only a few months or years, commercial aircraft can remain in service for decades. An early investment in smart automation can continue to provide cost and productivity savings ten or even twenty years down the road as a system evolves through early adoption, mainstream success and legacy support.

The comprehensive automation typical of **DevOps** is made possible by the wide-scale adoption of virtualization technologies. Virtualization uses software to emulate or simulate the hardware components of a system. With full virtualization, software written for the real hardware runs unmodified on the virtual system, even if the hardware and operating system underlying the virtual machine is different from the deployed hardware system.

**3** Virtualization has a wide variety of uses, from facilitating portability of legacy applications to new hardware, to enabling instrumentation and test automation that would not be possible with a real system.

Most of the discussion about virtualization in recent years has focused on its role in providing secure, independent and scale-able architectures for cloud computing.

Most SaaS software, as well as the backend of most mobile applications, now runs in the cloud, with services like Amazon AWS and Microsoft Azure dominating the market. However, virtual machines are also playing an important role in the development and maintenance of modern aerospace systems, and are integrated into a variety of avionics, flight management, radar and unmanned aerial vehicle applications.

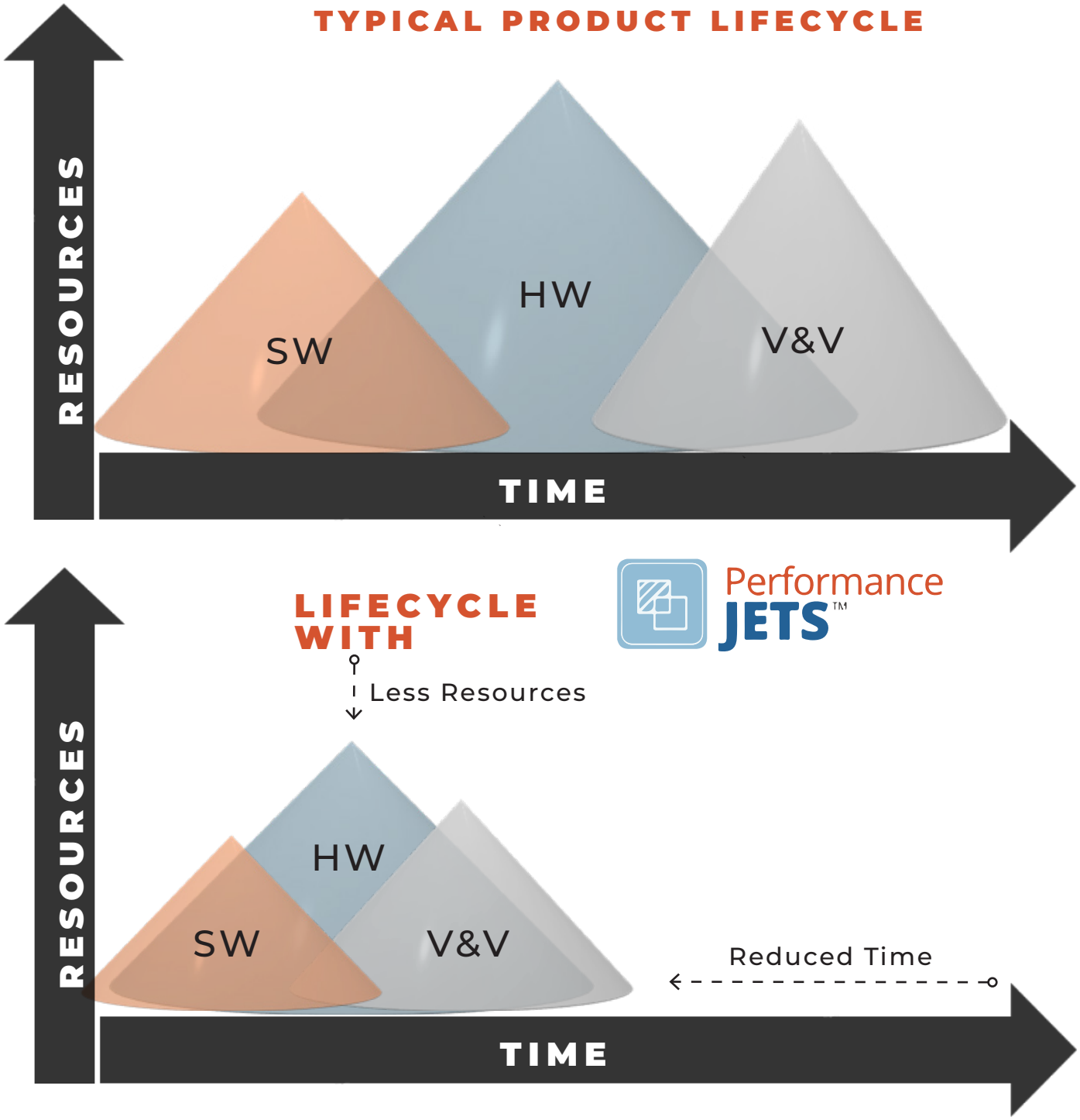


Figure 3: DevOps with JETS Can Reduce Development Cycle Time and Overall Costs

Performance Software is leading the transformation to **DevOps** for aerospace and defense, through their **JETS** virtualization technology.

**JETS** solves many of the problems that prevent aerospace programs from adopting **DevOps**, continuous delivery and large scale automation in their development practices. JETS is a full emulation and simulation platform, supporting a wide variety of embedded architectures and aerospace applications at the component, subsystem and system level.

**That's because each JETS implementation is designed with the specific component or system in mind: Performance Software engineers work with a hardware specification and design documentation to build a virtual environment that matches the target hardware and the hardware/software interface.**

This process yields a high-integrity prototype that can be used to further refine component design as well as begin application software development.

**4** Performance Software is leading the transformation to **DevOps** for aerospace and defense, through their **JETS** virtualization technology.

**JETS** has been designed to support a wide variety of underling architectures, including x86, x86-64, ARM, PPC, MIPS, SPARC, SH4, ETRAX CRIS and MicroBlaze, as well as a growing list of busses and hardware access schemes including Ethernet, SRIO, I2C, SPI, USB, PCI, RS232/422/485, A429, A664, Interrupt/ Memory Controllers and connected memory devices.

**JETS** emulation takes into account the complexities of multifarious real time systems by providing real and virtual clock timing and IO routing among multiple virtual machines. The virtualization platform can also be augmented by connecting line-replaceable units (LRUs) built from real system hardware, as well as integration of FPGAs and other custom hardware through an extensible Emulation Device Interface.

**JETS**' full virtualization provides a number of advantages.

**5** Binary compatibility for Virtualization technology reduces run-for score risk and cost."

As a result, **JETS** can take the place of real hardware for engineers writing, debugging and unit testing the code. Hardware test rigs are expensive, meaning they are often shared among multiple developers working on independent parts of the system, and often lack formal tools for deployment and management of the source baseline.

**With JETS, individual developers and small teams can each host their own instance or multiple instances of the component, even as the hardware continues to evolve.**

Teams employing virtualization can implement full API, feature and code coverage testing without resorting to simulators that provide lower fidelity. What's more, by building unit testing on virtual machines into the source code control and configuration management processes, the functional and performance impact of each change on the overall system quality can be measured within hours of the developer's submission, when the cost of incremental changes or bug fixes are at their absolute lowest.

Multiple platform configurations, usage scenarios and system states can even be stored as part of a test archive, with virtual machines dedicated to performing automated tests on a more-or-less continuous basis, as well as in response to developer check-ins, vendor patches and evolving system requirements.

**6** Through **JETS** Virtualization and **DevOps** integration, shorter development cycle with much greater overlap between hardware design, software development, integration and V&V testing are achieved.

Following a **DevOps** approach, systems enter the certification process sooner, and with higher overall quality.

**JETS** can also serve as a high-fidelity substitute for the real system in support of integration and system test. 'Delivery' to a system and system-of-systems test platform can mimic the continuous delivery of **DevOps** which has been proven to improve quality and reduce complexity and cost, even if ongoing delivery to the aircraft platform or LRU inventory is not practical due to certification and logistics concerns.

**JETS** supports bare metal debugging with the open-source GNU debugger (GDB) as well as debugging through an OS provided integrated development environment (IDE). Detailed logging and debugging as well as scripted test scenarios can be written using a single toolchain across platforms and components.

A variety of logging levels can be set, allowing introspection of register access, bus traffic, and memory usage. The debugger can be enabled from system start, so bootstrap code and firmware can be examined in the same manner as application code. **JETS** also provides a device tree view and non-halting access to memory contents. With these features, **JETS** can be configured to inject difficult-to-induce hardware errors into system testing, without compromising the integrity of real hardware or playing "what if" guessing games or conducting detailed failure analyses. As aviation failures often result from a cascading series of events, rather than a single fault, hardware fault injection offers the opportunity to significantly improve overall reliability and safety.

**JETS** can also be used as a virtual supplement to hardware. SaaS software applications often run in cloud computing instances systems. Public clouds like Amazon AWS and Microsoft Azure provide massively scale-able arrays of these virtual machines, which can be quickly installed and configured to support any operating system and application stack common to the PC computing world.

When application developers want to scale for testing, increased demand or operational maintenance, virtualization provides a quick and highly reliable solution. Such solutions have not been generally available to aerospace systems relying on custom embedded hardware.

**JETS solves this problem by facilitating seamless integration of real and virtual hardware. In fact, any mix of virtualization and hardware is possible as long as the appropriate Communication interfaces are provided. Real hardware and a JETS virtual machine can communicate via an IO converter. Thus a real A429 card can talk to an emulated A429 card, PC Ethernet can communicate to emulated Ethernet, and real A664 can be bridged to emulated A664.**

Single LRUs can be easily replicated to build up complex systems running similar hardware with different software solutions. In many instances, multiple LRUs can also be simulated within the same virtual environment, with fully virtualized IO, and then configured to cover a wide variety of test scenarios.

Using this approach, **JETS** virtualization can be scaled up to a full aircraft architecture. **JETS** even includes patented technology to synchronize event and input timings among virtual machines to improve the fidelity of real-time simulation. **JETS** can also be easily integrated into an existing virtualization or simulation framework in the same way real hardware would.

Finally, **JETS** fully supports FAA certification scenarios. The necessity for certification is one of the biggest obstacles to adopting **DevOps** in the aerospace and defense industry segments. As part of the virtualization development process, the Performance Software team will meet with your FAA

designated engineering representative (DER) to support approval for a percentage of "Run For Score" for your virtual platform.

This can be accomplished in a number of ways, including by having **JETS** certified as a test tool. By achieving this certification, teams can often receive equivalence credit for all application level software, since **JETS** emulation runs the target platform binaries unmodified.

**7** **JETS is currently providing significant benefit to a variety of billion-dollar aerospace and defense programs."**

**These include the Boeing 787, the Unmanned Aerial Vehicle (UAV), multiple business jet platforms and the Boeing 737.**

On the 787 platform, **JETS** was able to solve critical program issues related to the availability of debugging and test resources to team developers, the lack of available hardware for a critical subsystem component, and scale-ability challenges related to code coverage and test automation.

By shifting to a virtualization-based strategy, the 787 team was able to overcome the inherent challenges in testing and debugging embedded hardware resulting in cost-savings, improved quality, quicker execution, greater scale-ability and ultimately freeing developers allowing them to spend more time authoring quality code.

The Performance team has developed a comprehensive process for mentoring aerospace and defense engineering teams that want to take full advantage of virtualization and **DevOps** via **JETS**. Each project begins by identifying the minimum system requirements to implement virtualization. These include the SOC/processor and memory type to be emulated, how the devices interface with the processor (e.g. SPI, I2C, memory mapped, interrupts), data sheets for component interaction and expected device utilization.

Performance and scale-ability are then considered, together with additional requirements for large or complex processors. With these requirement in hand, Performance develops the virtual platform, trains users, and supports program integration. Throughout the process, performance brings its virtualization expertise so your development teams can focus on functionality, quality and efficient automation.

If you want to learn more about how Performance Software can help your organization transition to **DevOps**, improve quality and save time and cost, call **(623) 337-8240** or email **michael.johnson@psware.com**

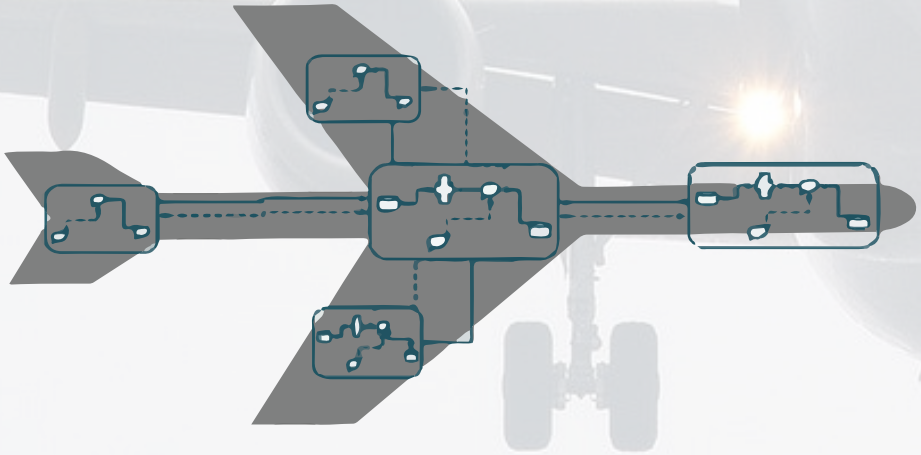

Figure 4: JETS Supports Scale-able Emulation of Full System-of-Systems Architecture


Performance
Seeing Things Differently