BY STEVE CARLSON, JAMES S.B. CHEW AND FRANK SCHIRRMEISTER

Developing Military Electronic Systems Calls For Holistic Strategy

■ One can view the Defense Department's Digital Modernization Strategy as a direct response to this 2018 National Defense Strategy goal: "Prototyping and experimentation should be used prior to defining requirements and commercial off-the-shelf systems. Platform electronics and software must be designed for routine replacement instead of static configurations that last more than a decade."

Within the strategy, one can argue that the Defense Department chief information officer's priorities — cybersecurity, artificial intelligence, cloud, command, control and communications — were developed to help achieve the aforementioned National Defense Strategy goal.

And that goes for the digital modernization goals as well: innovate for competitive advantage; optimize for efficiencies and improved capabilities; evolve cybersecurity for an agile and resilient defense posture; and cultivate talent for a ready digital workforce.

In response, the services are executing initiatives to meet these priorities and achieve these goals.

All these approaches are based on software development. Specifically, they are based on "DevSecOps," as is now being used with the software development approach. These are a set of practices that combine software development (Dev) and information-technology operations (Ops) with the aim to shorten the systems development lifecycle and provide continuous delivery with high software quality. When referenced as DevSecOps, the (Sec) acknowledges that for the Defense Department, security issues are of a paramount concern and must be addressed.

The use of DevOps has been a commercial best practice for years and it does address the department's desire for agility. In truth, its adoption of DevOps is an excellent first step.

But there is a reason why this is a popular joke about DevOps:

Question: "How do DevOps engineers change a lightbulb?"

Answer: "They don't. It's a hardware problem."

This joke highlights the wisdom of one of the popular quotes attributed to Alan Kay, the inventor of Smalltalk and the Alto, and the driving force behind Xerox PARC in 1982: "People who are really serious about software should make their own hardware."

Fifteen years ago, the commercial electronics ecosystem was organized as it had been since the 1970s. There were "chip guys," "software geeks" and "systems geniuses." Each group worked hard to optimize their craft and great gains were made, and sins were masked thanks to the all-powerful Moore's Law.

The limits of optimization began appearing somewhere between 2000 and 2005 with systems, software and single-core performance gains leveling off as predictions of power consumption causing rocket engine-level heat made waves in trade magazines. At that time, the first articles predicting the end of Moore's Law and the crucial co-dependency between software and hardware were published, with Herb Sutter's 2005 article in *Dr. Dobbs Journal*, "The Free Lunch Is Over" perhaps being the most prominent.

Sutter's article stated that microprocessor serial-processing speed is reaching its physical limit, leading to two main consequences.

First, processor manufacturers would have to focus on products that better support multi-threading such as multi-core processors. Second, software developers would be forced to develop massively multi-threaded programs as a way to better use such processors. The "free lunch" — the constant improvement of hardware performance that made a software developer's life easy — would come to an end.

Experts then predicted a new golden age of domain-specific architectures — custom hardware — and domain-specific languages: software optimized for the custom hardware.

The answer to ensure further gains was to optimize across the strata to support multi-core architectures. With this, the

problem of power became the key driver.

In particular, the burgeoning smartphone market saw battery life as a key limitation to adding new capabilities and a major source of customer dissatisfaction. Additionally, consumers were showing an appetite for features such as web browsing that demanded more and more processing power.

This all required specialized hardware support: networking, video, multi-tasking, graphics, low power, audio, security and camera.

All of these elements had been available in separate products, but never brought together in a phone. Each of them had been highly optimized. However, to make a phone with all of these features, very different success optimization metrics needed to be applied. The metrics required a new methodol-

ogy to optimize using a new cumulative objective function. To drive the cost function down, it became clear that the traditional serial strata of the system development process needed to be shattered and that the design chain had to be restructured.

For example, until the early 2000s, the design chain of embedded mobile systems was dominated by platform-based designs. The semiconductor vendors provided all drivers, and then interacted with software OS vendors, like Palm OS, Symbian, Microsoft Window CE and Pocket PC 2002 to port their operating systems to their silicon. They would then provide it jointly to device and equipment manufacturers. This situation has now been replaced by only two operating systems — iOS and Android — integrating all the required services as middleware in exchange to enable a much bigger ecosystem of apps that can be developed based on early representations of the hardware.

For example, think of the iOS and Android software development kits that are provided as pure software representations.

By taking on more responsibility for the hardware/software stack, a much bigger ecosystem of application developers has been unleashed. In exchange, however, the hardware abstraction layer, or HAL, of an Android device, for instance, must be architected and verified in a way that software development

iStock illustration

can start in parallel. This is what the industry today refers to as the "shift left."

Demolishing the barriers between teams begins with eliminating the serial hardware-then-software process. Instead, software development has, at least partially, "shifted left" to overlap with hardware design. This change often makes the software team a little uneasy at first, as working on a "squishy" hardware platform is new ground.

But, as the software developers begin to recognize that they do not have to work around all of the hardware bugs anymore, they now have a choice as to whether to fix the problem at the hardware source or work around it in software. In fact, once this hardware/software design process is implemented, many software developers relish the ability to push the problems back to the hardware team.

Furthermore, to really co-optimize hardware and software, the industry is entering a phase of custom, configurable hardware with associated software. One clear example of this trend is the emergence of programmable, extendable processor architectures, as well as a resurgence of reconfigurable architecture that can switch algorithms within very small timeframes.

It is worth noting that while a first wave of reconfigurable architectures was introduced in the early 2000s with long-forgotten startups like Adaptive Silicon, Elixent, Triscend, Morphics, Chameleon Systems, Quicksilver Technology and MathStar, they are finding a revival now with defense-specific programs.

The key benefit in system optimization of the shift-left trend is the early visibility of system size, weight and power characteristics. As for functional bugs, shift-left enables a choice of where to draw boundaries, and subsequently move them. That is, hardware-software tradeoff optimization is enabled for performance, power, thermal and reliability. Hardware emulation, a critical design automation technology required to make the shift-left a reality, can also often be combined with commercial virtual prototyping and "software-based emulation" based on open-source technologies like QEMU and VirtualBox.

All emulators are not equal. Accurate SWAP tradeoffs begin with accurate hardware representations. Register-transfer-level languages — such as VHDL, Verilog and SystemVerilog — can help with this. Hardware-software co-optimization methodologies require the accuracy of RTL hardware emulation.

To allow app development in a decoupled fashion, software-based emulation using technologies like QEMU and VirtualBox are often employed or provided in Android and iOS software development kits. The techniques that design teams choose to adopt depend on accuracy/performance/availability tradeoffs. Typically, the higher in the software stack the software to be developed resides, the more abstract the representations for development are.

The Defense Department's adoption of DevOps is an excellent first step, but it can't be the last. It's very tempting for some within the department to consider the adaptation of DevOps to be the easy fix to address their electronic system development, sustainment and modernization issues.

However, while we always hope for the easy fix — the one simple change that will erase a problem in a stroke — we all know that few things in life work this way.

Instead, success requires making 100 small steps go right — one after the other, no slipups, no goofs, everyone pitching in. Furthermore, we know that to truly solve a problem, one must tackle the root cause, not the effect. Hence, the rationale for the famous Alan Kay quote, "People who are really serious about software should make their own hardware."

To meet the intent of the 2018 National Defense Strategy goal for microelectronics, both hardware and software development must be addressed.

And the major lesson from the successful commercial electronics systems companies is this: If you really want good software for your system, you've got to have really good hardware that's been developed right alongside your software. **ND**

James S.B. Chew is group director, Frank Schirrmeister is senior group director of solutions marketing, and Steve Carlson is director of aerospace and defense solutions at Cadence Design Systems.