Vision, Reality, and Future of Software Defined Radios

David M. Tate and Lawrence N. Goeller

A commitment to hardware interface standards and modularity might allow DoD to realize at least some of the promise of the vision for military radios.

The Problem

The DoD vision for how software defined radios would revolutionize future military communications had a number of goals:

- **Be able to communicate (voice or data) between any two radios, once configured**
- **Be upgradable to use future communications systems on existing hardware**
- **Provide new and continuously improving networking capabilities over time**
- **Reduce future development costs for new wireless capabilities**
- **Reduce future procurement costs for radio hardware through higher volumes and longer useful radio lives**
- **Reduce future maintenance costs for radio hardware through commonality**

Reaching these goals is not feasible with current or near-term technology.

The invention of the software defined radio (SDR) in the early 1990s made it seem possible for the Department of Defense (DoD) to replace incrementally more than 100,000 diverse and incompatible radios with more versatile and interoperable radios.

The idea was that waveform software[1] would run as "applications" on generic radio hardware that would perform the necessary physical radio functions (e.g., carrier generation, modulation, synthesis, and multiplexing) as directed by the software. The analogy would be to various possible apps that can run on the same tablet or cell phone. The waveform software would be *portable* to new hardware with a minimum of effort.

In the late 1990s, DoD completed several demonstration projects and prototypes that seemed to confirm the viability

[1] In general, a waveform is the time series of radio signals that encodes information and is transmitted from one radio to another. The SDR community uses the term more broadly, to refer to all of the processing steps that encode and decode the information transmitted for a given mode of communications. These include modulation, channel encoding, multiplexing, frequency hopping, and so forth.

of the SDR approach for military communications. Based on this, the Programmable Modular Communications System (PMCS) and Joint Tactical Radio System (JTRS) programs were launched and eventually merged under the JTRS label. The JTRS project was broken out into a number of separate programs—one for waveform software, one for Army vehicular radios, one for small form-factor radios, and so on. In the end, none of these programs delivered the envisioned capability of portable software that could run on multiple physical radios while delivering satisfactory operational performance.

The Director, Performance Assessment and Root Cause Analysis (PARCA) asked IDA to investigate the root causes behind this set of failed development programs. In particular, PARCA was interested in whether there were technical barriers to success that were not recognized at the time. The JTRS programs suffered from many confounding difficulties, including requirements changes, an awkward program structure and acquisition plan, and perverse contractor incentives. In addition to these, we were able to identify technical issues that alone would have been sufficient to prevent program success.

The problem has to do with the different kinds of hardware used to do computing. The original concept for portable SDR implicitly assumed that the software implementing any waveform would be executed on general purpose processors (GPPs) of the kind familiar to personal computer owners. GPPs are extremely flexible in the kinds of logic and computation they can implement, and can be programmed in high-level languages that allow the same code to run on many different machines. A standard—the Software Communications Architecture (SCA)—was developed to describe how the waveform software would interact with the hardware controlling the radio functions. The SCA specified two particular commercial middleware solutions: the Common Object Request Broker Architecture (CORBA) to mediate between software commands and radio function execution, and the Portable Operating System Interface (POSIX) application programming interface for real-time control requirements. Waveform developers would thus be able to write high-level programming language function calls that CORBA and POSIX would translate into hardware actions specific to the radio hardware being used.

As it turned out, current (and near-term foreseeable) GPPs simply could not provide the computational performance required to implement waveforms within the practical operating constraints of a military radio. The CORBA and POSIX calls add significant computational overhead to each action, so that powerful processors are needed to keep up with the real-time demands of radio frequency signal processing. Any processor powerful enough to achieve the needed performance could not meet the strict military limits on physical size and weight, heat generation, battery life, transmission range, and so forth.

In response to these difficulties, the hardware developers turned to Field-Programmable Gate Array (FPGA) technology as a compromise between

the ease of coding and portability that GPPs would have provided and the performance requirements of the military radios. FPGAs are somewhat programmable (though less flexible than GPPs), but are faster and draw less power than GPPs. The hope was that this middle ground would allow the radios to meet both performance and form-factor requirements, while preserving some portability of waveform software.

Unfortunately, FPGAs proved not to be a "sweet spot" between GPPs and dedicated signal processing hardware of the sort used in legacy radios and cellular telephones. Even using the latest "system on a chip" generation of FPGAs, the radios were still generally unacceptable in range, weight, heat generation, and waveform performance. Adding to the problem, the SCA specification was not defined at a level detailed enough to guarantee compatibility with other hardware, or portability of FPGA code. The JTRS programs thus lost the benefits of easily porting waveform software from one platform to another, without solving the performance and form-factor issues.

In the end, SCA became part of the problem, rather than part of the solution. There is no current technology for which SCA is sufficient to ensure portability and achieve the necessary real-time performance for military wireless communications. On FPGA-based systems, requiring SCA software compliance hinders radio performance and increases waveform development costs, while providing no compensating benefits.

Even if computing power continues to increase exponentially according to "Moore's Law," it will be many years before GPPs will be small, fast, cheap, and efficient enough to support SCA-based SDR—and even then it will still be likely that computer hardware processing capabilities will continue to improve more quickly than software capabilities. In that case, it seems unlikely that anyone would want to install multiple generations of waveform software on the same piece of hardware, when the hardware will become obsolete more quickly than the software.

Having identified this fundamental barrier to the vision of SDR-enabled waveform portability, we propose a potential alternative approach that might recover some of the sought benefits. That approach uses a different analogy—instead of thinking of waveforms as being like software applications on a tablet or phone, think of them as hardware peripherals, like graphics cards or hard drives used by a personal computer. Graphics card capabilities change relatively rapidly, and it is not uncommon for high-end users (such as video editors or serious computer gamers) to change graphics cards several times before buying a new computer motherboard.

In this approach, the core radio hardware would be the "motherboard," overdesigned for the hardware capabilities available at its inception. The radio would provide a standard interface bus for radio function modules—analog-to-digital and digital-to-analog converters, power amplifiers,

antennas, cryptographic processors, user interfaces, network routers, and (especially) waveform synthesizers. Two radios would become interoperable by plugging in the same waveform "card" (and the appropriate other modules). Modules would thus be portable and reusable, but could be replaced relatively cheaply as new and more capable versions were developed. The stressing processor functions for radio performance could be upgraded without buying a whole new radio. The standard for the radio interface bus would be open, but the workings of the individual radio function modules could be proprietary, with no loss of portability or interoperability.

Hardware continues to improve much more quickly than software. A modernization approach in which hardware would stay in service for years or decades, upgraded only via software, would contrast sharply with the path that consumer electronics have taken. A commitment to hardware interface standards and modularity might, however, allow DoD to realize at least some of the promise of the vision for military radios. In the meantime, the SCA has become a barrier to progress within the world of military wireless communications.

Dr. Tate *is a Research Staff Member in IDA's Cost Analysis and Research Division. He holds a Doctor of Philosophy in operations research from Cornell University.*

Dr. Goeller *is a Research Staff Member in IDA's Cost Analysis and Research Division. He holds a Doctor of Philosophy in physics from Rice University.*

The original article was published in the *Institute of Electrical and Electronics Engineers (IEEE) Military Communications Conference (MILCOM) Proceedings*, November 2014.

"A Technical Review of Software Defined Radios: Vision, Reality, and Current Status"

http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6956963