

## IS THE WIRELESS INDUSTRY READY FOR COMPONENT BASED SYSTEM DEVELOPMENT?

Lee Pucker

Spectrum Signal Processing, Inc.

Imagine, if you will, the following scenario:

*You are a communications systems engineer responsible for the creation of a subsystem for a new wireless product. This subsystem will include all of the hardware and software necessary to support the physical, data link, and network layer processing for the new product, and toward that end marketing and the system architect have provided you with one or more waveform specifications defining these layers, as well as your budget for size, weight, power, and unit cost.*

*You begin your task by working with your waveform development team to create a component model detailing the various processing and control blocks required to support each waveform specification. This model is built using an integrated development environment (IDE) hosting a library of industry-standard component definitions, with each component in the library parameterized to allow it to be optimized for a specific application. Key elements of the component models are the use cases defining the functional operation of the processing and control blocks supporting the various modes of operation for the target waveform, and the associated system sequence diagrams illustrating the required interactions between components to meet the overall waveform requirements. For those components that are not available in the library, you generate a new component model from a defined selection of industry-standard component primitives.*

*Once the prototype component models are complete, your waveform development team uses the simulation capabilities of the IDE to perform functional simulations of the waveform processing to select one of the predefined signal processing algorithms associated with each component block. As a part of this process, you create a waveform test bench in the IDE emulating the operating environments of the component models. Based on this functional simulation, the waveform development team can tweak the component models, changing the parameters of each component block to optimize performance in the simulated environment or even replacing whole component assemblies to improve the overall performance.*

*With a working platform-independent component model in hand, you now work with the project's hardware and software leads on creating deployment models for your waveform applications, mapping the defined waveform component blocks to specific processing devices and optimizing to meet your size, weight, power and cost parameters. To assist you with this, the IDE provides you with optimized component libraries from the various chip manufacturers with which your company works, mapping the industry-standard component blocks to their respective devices, and providing benchmarks on expected performance and resource utilization. These processor-specific component models are defined based on working application code and use industry-standard mechanisms for*

*enabling communications with other components operating on the same signal processing device, as well as components operating on remote devices utilizing the candidate chips input/output interfaces. These mechanisms are overlaid in your IDE with the industry-standard application framework supported by your company, which defines how the various components are set up, torn down, configured, and controlled within your company's portfolio of products.*

*If support for multiple waveforms is required for the subsystem, your team can use the IDE to quickly explore associated trade-offs in device selection, optimizing the architecture for the worst case waveform requirements while minimizing size, weight, power, and cost. If multiple deployment models are possible, the IDE can be used to quickly generate each valid model, with the output from the IDE vetted with marketing and the systems architect to define the most appropriate approach going forward.*

*Once the team is satisfied that the deployment models meet all the specified requirements, you click the generate button in your IDE. This generates skeleton hardware and software requirements specifications, allowing your engineering teams to quickly prototype the signal processing platform based on the deployment models. In addition, the application code for all of the waveform components is generated, including Hardware Description Language (HDL) where appropriate, as well as the setup and control code necessary to instantiate the waveform on the signal processing subsystem. Finally, an itemized list of third-party IP is generated, allowing your marketing team to access the licensing cost associated with the waveform applications.*

Component-based development such as this has long been the Holy Grail of signal processing system design for many communications systems engineers. The reason for this is simple: the existence of an integrated development environment as defined above has the potential to shave months or even years off the time necessary to move from specification to field trial over a more traditional development model. As such, this development model could represent significant savings in both development costs and time to market for new wireless communications systems. These savings affect not only systems targeted for production, but also internal research and development programs exploring new and innovative wireless techniques that could be used to differentiate a company in the market. These programs can use the development model to quickly prototype technologies such as multi-user detection or smart antennas to demonstrate their efficacy in an actual field environment.

There are a number of tools available that support pieces of the solution necessary in an integrated environment for component-based development: just Google *DSP radio "design environment" "block diagram"* and look at the results you get. Even so, we are still a few years away from achieving the idealistic model described above, primarily because of a lack of mature standards that are broadly accepted across the wireless

industry in the following four areas.

**Common Modeling Language** — An important part of creating an integrated environment supporting component-based development is having a common modeling language supported throughout the environment that allows the disparate development teams involved in wireless systems design to communicate in a common manner. Two modeling languages seem to dominate the wireless industry: the Specifications and Description Language (SDL) maintained as a standard by the International Telecommunication Union (ITU) and the Unified Modeling Language (UML) maintained by the Object Management Group (OMG) [1, 2]. Both of these languages are widely used throughout the industry, but until recently SDL was considered the language of choice for real-time systems because of deficiencies in the UML specification. This has since changed, and UML v. 2.0 adds many of the features necessary to support the required real-time modeling [3]. Ultimately, some combination of SDL and UML may define the modeling environment of choice in the wireless industry [4].

It should be noted that the OMG Software-Based Communications Domain Task Force has developed a platform-independent model (PIM) for software radio components that could act as a key element in defining a common modeling language [5]. This model was developed in cooperation with the U.S. Joint Tactical Radio System (JTRS) program, and is built following the OMG's Model-Driven Architecture (MDA®) [6, 7].

**Standard Waveform Functional Blocks** — A key element of the idealized model is the ability of the tools to allow the engineering team to choose whether to deploy, for example, a quaternary phase shift keying (QPSK) demodulator on a digital signal processor (DSP) using one vendor's implementation or on a field programmable gate array (FPGA) using another vendor's implementation based on the resource utilization and performance implications associated with the selection. This requires that both vendors develop their QPSK demodulators following the same component standard, ensuring that the demodulator "looks" the same to the other components deployed in the system regardless of which deployment model is selected. A library of these standardized blocks, as well as primitives that can be used to generate new blocks, must be defined within the industry in some detail, perhaps with different libraries necessary in varying market segments.

**Common Hardware Abstraction** — Actualization of a platform-independent component model into deployable code that can operate on a specific device and is guaranteed to interoperate with other deployed components, regardless of where those components are located within the system, requires a standard model for encapsulating the IP associated with the component in a manner that is consistent from program to program and vendor to vendor. To date, there are a number of device-centric component models available in the industry that define a basis for supporting component encapsulation at this level. These models include:

- General-purpose processor: CORBA Component Model, J2EE Enterprise JavaBeans [8, 9]
- Digital signal processing devices: XDAIS [10]
- Reconfigurable logic/system on a chip: Open Cores Protocol, WishBone, AMBA [11–13]

A key element that must be addressed by industry to achieve the idealized model is interoperability between these device-specific component models. Work by programs such as the End-to-End Reconfigurability (E2R) program in the com-

mercial space and the JTRS in the defense arena may define the standards of choice in specific segments and the rules supporting interoperability between device standards [14, 15].

**Standard Application Frameworks** — A standard application framework provides a mechanism for deploying waveform components onto signal processing devices that vary from project to project and vendor to vendor. This allows the waveform functional blocks to be defined with the necessary hooks for setup, teardown, configuration, and control that are common across the industry. The JTRS Software Communications Architecture's core framework is among the most mature application frameworks addressing these needs, with an architecture that has been validated through multiple implementations [16]. Other programs, such as the IST-SCOUT project, have attempted to define an application framework for terminal management that addresses the specific needs of the commercial space, and this area is being actively explored as a part of E2R [17, 18]. Ultimately, multiple application frameworks may be appropriate or even required to address the specific needs of the various segments within the wireless industry, and as such the tools will have to allow support.

These four elements must mature, either industry-wide or by market segment, to allow for the creation of an integrated environment supporting component-based development as described above. There is good reason to believe that over the next several years this will occur, with programs such as the JTRS and E2R leading the way in driving standardization of the requisite technologies. Work such as this can then be leveraged by industry through groups such as the SDR Forum's Design Flow and Tools Working Group in moving toward the idealized model [19].

## REFERENCES

- [1] "What Is SDL?" <http://www.sdl-forum.org/SDL/index.htm>
- [2] "UML Resource Page," <http://www.uml.org/>
- [3] M. Bjorkander and C. Kobryn, "Architecting Systems with UML 2.0," *IEEE Software*, July 2003
- [4] M. Chetty, "A Comparison of Unified Modeling Language (UML) and Specification and Description Language (SDL)," [http://pubs.cs.uct.ac.za/archive/00000054/01/uml\\_sdl.pdf](http://pubs.cs.uct.ac.za/archive/00000054/01/uml_sdl.pdf)
- [5] OMG, "PIM and PSM For Software Radio Components," <http://www.omg.org/docs/dtc/04-05-04.pdf>
- [6] U.S. Army Public Affairs, "Joint Tactical Radio System (JTRS) Architecture Is the Basis for International Commercial Standards," Washington, DC, <http://jtrs.army.mil/sections/news/releases/OMG-based-on-SCA.html>
- [7] OMG, "MDA Specifications," <http://www.omg.org/mda/specs.htm>
- [8] OMG, "CORBA Component Model V3.0," <http://www.omg.org/technology/documents/formal/components.htm>
- [9] J2EE Enterprise JavaBeans Technology, <http://java.sun.com/products/ejb/index.jsp>
- [10] Texas Instruments, "TMS320™ DSP Algorithm Standard," <http://focus.ti.com/lit/ml/sprt254a/sprt254a.pdf>
- [11] OCP Int'l. Partnership, "Open Cores Protocol," [http://www.ocpip.org/socket/datasheets/OCP\\_Datasheet.pdf](http://www.ocpip.org/socket/datasheets/OCP_Datasheet.pdf)
- [12] Opencores.org, "SoC Interconnection: Wishbone," <http://www.opencores.org/projects/cgi/web/wishbone/wishbone>
- [13] AMBA, <http://www.arm.com/products/solutions/AMBAHomePage.html>
- [14] J. Brakensiek, Ed., "E2R Deliverable D4.2: State of the Art and Outlook," [http://e2r.motlabs.com/Deliverables/E2R\\_WP4\\_D4.2\\_040723.pdf](http://e2r.motlabs.com/Deliverables/E2R_WP4_D4.2_040723.pdf)
- [15] JTRS Joint Program Office, "Specialized Hardware Supplement to the Software Communication Architecture Specification," [http://jtrs.army.mil/documents/sca\\_documents/V3.0/SCA/Specialized/Hardware/Supplement/3.0.pdf](http://jtrs.army.mil/documents/sca_documents/V3.0/SCA/Specialized/Hardware/Supplement/3.0.pdf)
- [16] Raytheon, "Raytheon Consortium Validates JTRS Architecture; Receives \$4.6 Million to Continue JTRS Architecture Support," <http://www.raytheon.com/newsroom/briefs/jtrssa.html>
- [17] SCOUT, "Requirements for Software Methodologies for Reconfigurability Such as Agents, Middleware and Adaptive Protocols," <http://www.ist-scout.org>
- [18] Stavroulaki et al., "A System for the Management and Control of Equipment in an End-to-End Reconfigurability Context," [http://e2r.motlabs.com/dissemination/conferences/E2R\\_WVRF12\\_WG6\\_8.pdf](http://e2r.motlabs.com/dissemination/conferences/E2R_WVRF12_WG6_8.pdf)
- [19] L. Pucker, "Design Process and Tools Workinggroup Charter," [http://www.sdrforum.org/MTGS/mtg\\_42\\_jan05/05\\_i\\_0012\\_v0\\_00\\_dptwg\\_charter\\_01\\_20\\_05.pdf](http://www.sdrforum.org/MTGS/mtg_42_jan05/05_i_0012_v0_00_dptwg_charter_01_20_05.pdf)