

Software defined radios – software aspects and the future (2)

Part 1 of this article in the previous issue of News from Rohde & Schwarz (No. 182) discussed the effect of the software defined radio concept on radio equipment architecture. This follow-up article examines the effect on software structure and the software development process. It also provides a brief look at the future.

A forward-looking concept

By definition, all main tasks in software defined radios (SDRs) are software-dependent. In the receive path, these tasks include analog/digital conversion, downconversion (by digital downconverters), intermediate frequency filtering, demodulation, decoding and voice signal processing. The reciprocal tasks are performed in the transmit path, plus additional tasks such as linearization of the transmitter output stage.

Increasingly, encryption and other security functions are also handled by software on processors suitable for this purpose. In the past, such functions were largely relevant only for military applications. Today, however, they are of great importance in the civil sector as well.

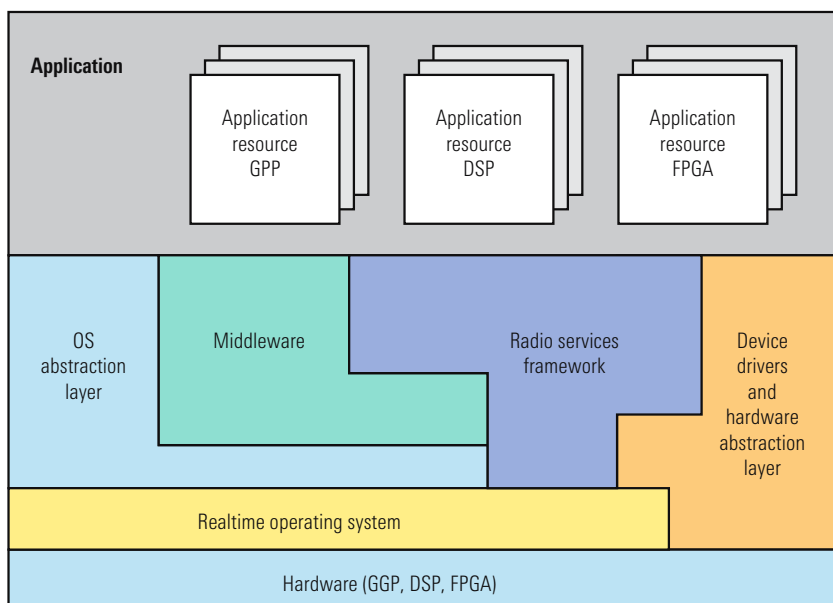
Other SDR tasks that often go unnoticed include the calibration and monitoring of the radio and the provision of user interfaces and remote control.

SDRs also involve various tasks that are relevant to internal organization and sequence control. A primary example is resource management, which includes assigning memory and managing computing power when configurations are changed.

New directions in software development

SDR software must be planned and developed with design goals in mind. The ultimate objective is to achieve and optimize short development times, low development costs, compliance with specifications, low manufacturing costs, etc.

FIG 1 Software architecture in SDRs with abstraction layers.



Experience has shown that optimizing design goals solely within a single development project is not sufficient. This is made clear especially by the fact that software has a significantly longer life-span than hardware. Thus, it makes economic sense to create reusable software. But software is reusable only if it is relatively easy to port and modify. In other words, you must be able to transfer it between platforms that may have very different processors and data transport architectures. In addition, you must be able to integrate new customer requirements quickly and easily. As a rule of thumb, porting costs must not exceed 10% to 20% of the original development costs.

One method frequently used in computer science to reduce porting costs is the abstraction of functionalities. In this method, a software application (e.g. software for generating a waveform in the air) does not directly access the hardware or the operating system. Instead, it communicates via an intermediate layer (abstraction layer) that describes a generalized model of the functionality made available by the hardware. If the application program interfaces (APIs) of the intermediate layer have been standardized, the functionality can be addressed in the same way on any platform. FIG 1 shows this principle, which is also used in radios from Rohde & Schwarz.

The Software Communications Architecture (SCA) of the Joint Tactical Radio System (JTRS) initiative of the US Department of Defense is an important step toward standardizing software interfaces and thus toward improving software portability. At least some parts of the SCA are also expected to find use in civil applications. A brief description of the SCA is provided in the box on page 55.

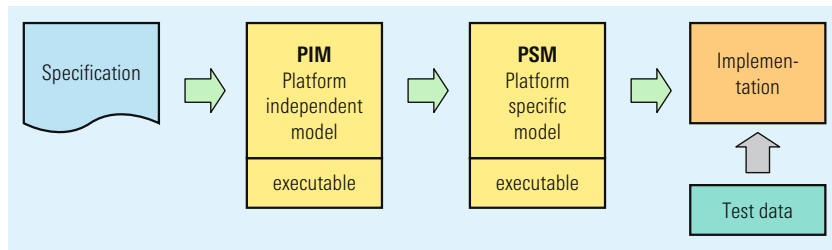


FIG 2 Software development process.

Model Driven Architecture

Since software can be highly complex and since it needs to be reusable, you must apply a structured approach (“process”) when developing it. By developing the Model Driven Architecture (MDA), the Object Management Group (OMG; see part 1 of this article) has standardized a software development process that is extremely valuable for developing not only SDRs. This process provides multiple models that build upon each other (FIG 2).

First, you create the specification and use it to develop the platform independent model (PIM), which defines the software functionalities independent of the implementation. In some cases, you also develop an executable version of the PIM. You can do this by using tools such as UML (unified modelling language), MATLAB® und Simulink®.

The next step is to create the platform specific model (PSM). In this model, the software is partitioned to various computing components such as GPPs, DSPs or FPGAs – while taking into account a model for the middleware. You can also produce an executable version of the PSM. Automatic generation of code from the PSM is also a good feature to have, since it allows you to create components that can serve as the basis for implementation. The objective is to reduce manual code generation in the implementation phase to a minimum.

The last step in creating the product is the test phase, which is for verifying that the software adheres to specifications. This requires developing test strategies, defining test stimuli and automating the tests to the extent possible.

New tools

To write software for SDRs, you need to use development tools. When you are developing waveform software, you need these tools even when defining the waveform to be developed. In addition to the standard tools already available, description languages and concepts are being developed that will probably mature over the next few years and then find wide application. Two examples are the waveform description language WDL and the radio description language RDL (The VANU Radio Description Language™). The underlying principle of these languages and the corresponding waveform development tools is to use components (for specific modulation and coding methods or protocols, for example) to create the required waveform.

Added value through software downloads

A significant benefit of SDRs is the broad flexibility made possible through software downloading. Software downloading is the capability of a radio system to significantly change the char-

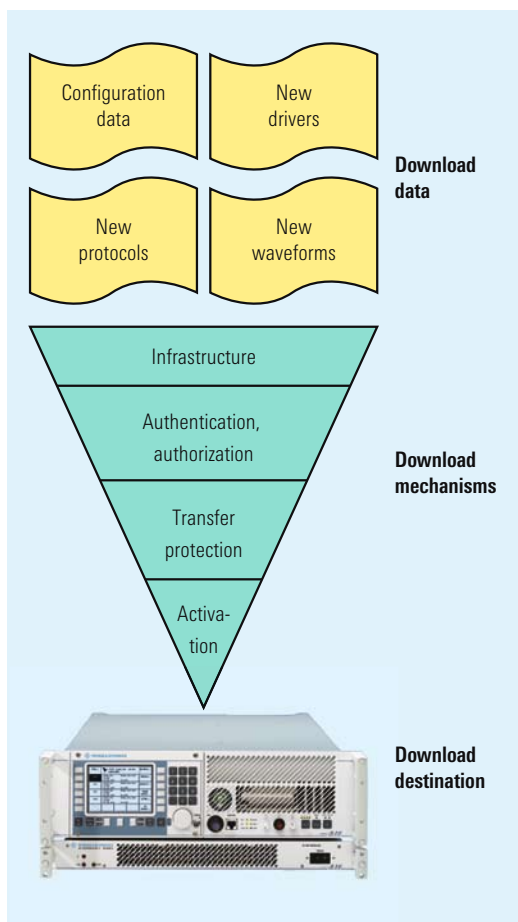


FIG 3 Mechanisms and objects for software download.

- Characteristics of a radio by transferring programs, parts of programs and data (FIG 3). Examples include updating parameter tables for handoff algorithms, updating driver software and expanding functionality by downloading a software package containing a new waveform.

Of course, the usability of each download varies from application to application. For example, the requirements and permissible solutions specific to a remote military radio station or a mobile radio base station will be different from those for a mobile phone, which has special requirements for power consumption and manufacturing costs.

There are two basic types of software downloads. Their main difference is how

data is transferred to the radio. Local downloads are performed on-site, e.g. from a laptop via a USB interface or from a server via a LAN. In contrast, wireless remote downloads are performed via the radio's air interface. Both download types require that suitable mechanisms be available in order to transfer data to the destination in a secure manner. The first criterion is having the infrastructure that provides the link for transferring data from the control center to the radio. The download must then be authenticated to ensure that data is from an authorized source. Since the data to be transferred is frequently executable code, ensuring error-free transmission is also necessary. This is achieved by using special codings and transmission protocols.

Trends and advances

SDRs will continue to penetrate markets for radios worldwide. In the civil sector, the leading market will definitely continue to be the market for base stations. Users are highly interested in software downloads since downloads mean cost savings due to the large number of transfer methods, wide distribution of stations and the frequent need to update and modify software. In contrast, the concept of a pure SDR will catch on less quickly in the area of mobile phones. These devices place extremely high requirements on miniaturization, manufacturing costs and power consumption, and they have a short lifespan. Yet, software downloads are also of commercial interest here, which means that certain aspects of SDRs will also come into use at an early stage.

The first generation of SDRs consisted of early developments of software-based products and demo models resulting from various research programs. The second generation of SDRs is already on the market (e.g. mobile

radio base stations or the Software Defined Radio Family R&S®M3xR from Rohde & Schwarz). The third generation is now in development and will implement advanced technologies in the hardware and software.

Improvements in analog/digital converters will make it possible to increase an instrument's dynamic range and sampling rate, thus allowing you to shift the point of A/D conversion closer to the antenna. However, even these instruments will also contain analog components in the frontend.

According to Moore's Law, which will remain valid in the foreseeable future, the available computing power of components will rise continuously and significantly. Thus, improvements in the bandwidth and quality of the signal processing section are only a matter of time. In addition, new types of components such as micro-electromechanical systems (MEMS) and super-conducting components will become available as off-the-shelf products. MEMS enable you to implement switches and filters with technology used in the production of semiconductors (application of material layers, etching, etc). Such switches and filters have characteristics that cannot be achieved with conventional methods. Superconducting elements take advantage of the extremely low losses, e.g. when it comes to implementing filters of extremely high quality or to implementing tunnel effects for very fast switches (Josephson elements).

With regard to software, third-generation SDRs will significantly improve the portability of radio software as a result of internationally coordinated architectures. An important point is the standardization of the operating systems, the middleware layers and the APIs of the functional units. An important milestone in this direction is the previously mentioned SCA of the JTRS initiative and

the related OMG industry standards, e.g. Minimum CORBA™ und SWRADIO.

The second generation of SDRs is already on the market and very successful. The use of new technologies – both in hardware and in software – will lead to the gradual transition to the third generation. For users, the most important factor will be how industry translates experience obtained during the development of the second generation into innovative solutions for the third generation.

Dr Rüdiger Leschhorn;
Dr Boyd Buchin

Software Communications Architecture (SCA)

FIG 4 shows the architecture of a radio designed in line with the SCA standard. The SCA is based on five principles. First, the interfaces to the operating system are standardized. A subset of the POSIX standard, which is derived from UNIX, is defined. The applications (e.g. waveform software) are allowed to use only this subset for system accesses. Second, CORBA™, which is an objected-oriented communications standard for distributed systems, is generally used for communications between software compo-

nents. Third, the mechanisms for loading and changing waveforms and other applications are precisely defined. The instance that handles this task is the core framework. Fourth, the manner in which the radio-specific hardware is mapped in the software is defined. This uniform abstraction of the hardware is called the hardware abstraction layer (HAL). Fifth, the rough modularization of the software applications is specified, and even the APIs for some of the software components are defined.

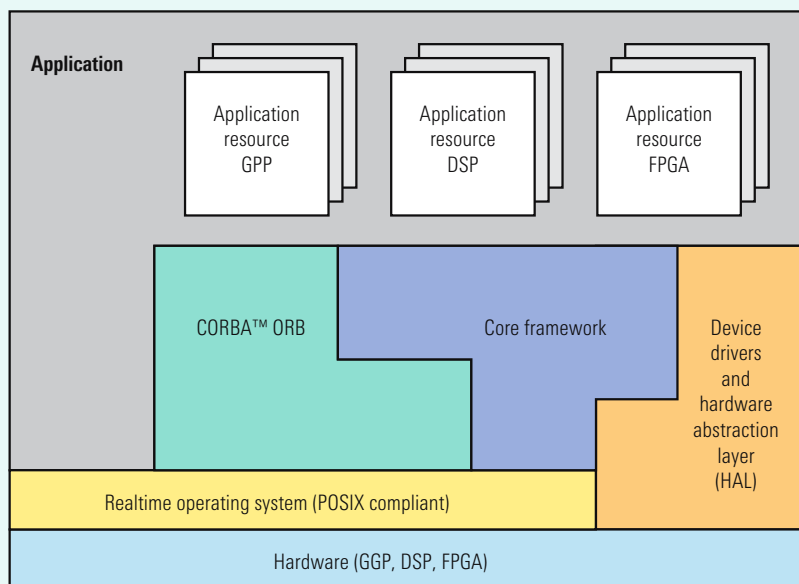


FIG 4 Architecture of a JTRS/SCA radio (simplified).