SOFTWARE RADIO ARCHITECTURE

Object-Oriented Approaches to Wireless Systems Engineering

Joseph Mitola III
Designations used by companies to distinguish their products are often claimed as trademarks. In all instances where John Wiley & Sons, Inc., is aware of a claim, the product names appear in initial capital or ALL CAPITAL LETTERS. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

Copyright © 2000 by John Wiley & Sons, Inc. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic or mechanical, including uploading, downloading, printing, decompiling, recording or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without the prior written permission of the Publisher. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 605 Third Avenue, New York, NY 10158-0012, (212) 850-6011, fax (212) 850-6008, E-Mail: PERMREQ @ WILEY.COM.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold with the understanding that the publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional person should be sought.


This title is also available in print as ISBN 0-471-38492-5

For more information about Wiley products, visit our web site at www.Wiley.com.
1 Introduction and Overview

I. REVOLUTION AND EVOLUTION

We are now in the midst of another revolution in radio systems engineering. Throughout the 1970s and 1980s, radio systems migrated from analog to digital in almost every respect from system control to source and channel coding to hardware technology. In the early 1990s, the software radio revolution began to extend these horizons by liberating radio-based services from chronic dependency on hard-wired characteristics of the radio, including:

- Radio frequency (RF) band
- RF channel bandwidth and coding
- Propagation media access
- Link layer protocols

Today the evolution toward practical software radios is accelerating through a combination of techniques. These include smart antennas, multiband antennas, and wideband RF devices. Wideband analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) access GHz of spectrum instantaneously. IF, baseband, and bitstream processing is implemented in increasingly general-purpose programmable processors. The resulting software-defined radio (SDR) extends the evolution of programmable hardware, increasing flexibility via increased programmability. The ideal software radio (SWR) represents the point of maximum flexibility in this evolution. In part, the software radio is an ideal that may never be fully implemented. The principles of the software radio nevertheless illuminate tradeoffs among radio architectures. SDR implementations “future-proof” infrastructure against continually evolving standards. Software radio architecture permits one to insert SDR technology gracefully and affordably. For a clear path for product evolution, one must understand the contributions of the ideal software radio to a specific application or market niche. The attempts of researchers to build ideal software radios yield lessons learned from these technology pathfinders. This text assembles these lessons into a coherent process for defining and evolving software radio architecture. It includes insights necessary to invest wisely in SDR-enabling technology. More importantly, it assembles the foundation on which those pursuing this technology can establish a software-radio systems-engineering process through which to navi-
gate the dangerous shoals of this revolutionary evolution of radio engineering.

II. A SYSTEMATIC EXPOSITION

This text first introduces the fundamental concepts of the software radio. These include the placement of the ADC near the antenna, the criticality of real-time streams, and the mix of implementation alternatives from baseband DSP through a variety of SDR alternatives. It then establishes the commercial and military drivers for an open-architecture for software-defined radios. Before addressing subsystem architectures, it identifies the aspects of the radio systems architecture that drive complexity. This is essential because SDR projects often fail because of unanticipated software complexity. It then covers the architecture principles by subsystem from antenna and RF conversion through DSP and software. It completes the core technical discussion by showing how to balance software computational demand against hardware processing capacity to produce software radios that meet specifications, on time and within budget. The text concludes with an overview of applications, including smart antennas and a mobile disaster-relief case study.

This first chapter provides an overview of the software radio (r)evolution shaping wireless systems engineering today. It introduces the software radio functional architecture. It also explains in more detail how analog, digital, and software radios form an implementation continuum, the software radio phase space. After completing the program of study represented by this text, a top-notch systems engineer will be able to position each project in implementation space as the technical, risk, and economic needs of the application dictate. The goal is to introduce most of the new concepts presented in this text.

It is worth emphasizing that this book does not try to sell the software radio. On the contrary, a software radio approach sometimes yields an ineffective product. One must fully appreciate how analog, digital, and software-intensive approaches complement each other. One may then understand the advantages and pitfalls of each. Ultimately, the reader should be able to decide when, where, and how to apply software radio technology. Thus empowered, each of the many participants in the software radio architecture (r)evolution will be able to contribute with greater impact.

III. THE IDEAL SOFTWARE RADIO

This section presents a top-down approach to the software radio architecture. The top level components of an ideal software radio handset consist of a power supply, an antenna, a multiband RF converter, and a single chip containing ADC and DAC. The on-chip general-purpose processor and memory that perform the radio functions are illustrated in Figure 1-1.
The generic mobile software radio terminal interfaces directly to the user (e.g., via voice, data, fax, and/or multimedia) and to the air interface. Driven by convenience and battery life, the mobile unit minimizes dissipated power and manufacturing parts count by maximizing hardware integration. The generic base station interfaces to the air and to the Public Switched Telephone Network (PSTN). With access to the power grid, base stations may employ modular, open-architecture hardware that facilitates technology insertion. Technology insertion opportunities future-proof the wireless infrastructure against the inevitable continuing evolution of air interfaces. Fully instrumented base stations support operations, administration, and maintenance (OA&M), while engineers and researchers may access the SDR network via services development workstation(s). Military base stations (nodes) need to support multiple networks on multiple RF bands with multiple air interfaces (modes). Such base stations may be formed by the co-location of diverse radios on mobile vehicles. These configurations often interfere with each other. The military calls this “cosite interference.” The software radio base station attempting to support traffic on multiple channels in the same band can generate self-interference unless transmissions are coordinated or interference is actively cancelled.

The placement of the ADC and DAC as close to the antenna as possible and the definition of radio functions in software are the hallmarks of the software radio. Software radio mobile units and base stations share a common software factory that downloads personalities to the mobile units and updates to the infrastructure. Thus, although software radios use digital techniques, software-
controlled digital radios are not necessarily software radios. The key difference is the total programmability of software radios, including programmable RF bands, channel access modes, and channel modulation.

SDR designs use Application-Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs), Digital Signal Processors (DSPs), and general purpose (GP) processor technologies. SDR has become practical as costs per millions of instructions per second (MIPS) of DSPs and general-purpose central processor units (CPUs) have dropped below $10 per MIPS. The economics of software radios become increasingly compelling as demands for flexibility increase while these costs continue to drop by a factor of two every few years. At the same time, absolute processing capacities continue to climb into the hundreds of millions of floating-point operations per second (MFLOPS) to billions of FLOPS (GFLOPS) per chip. At this point, software radio technology can cost-effectively implement commercial first-generation (1G) analog and second-generation (2G) digital mobile cellular radio air interfaces. Over time, wideband third generation (3G) air interfaces will also yield to software techniques on wideband RF platforms. In the interim, SDR implementations will require a mix of hardware-intensive techniques such as ASICs.

In addition, ADCs and DACs available in low-cost chips and single-board open-architecture configurations offer bandwidths of tens of MHz with the dynamic range required for software radio applications. Multimedia requirements for desktop and wireless personal digital assistants (PDAs) continue to exert downward pressure on parts count and on power consumption of such chip sets. This trend will push the ideal software radio technology from the base station to the mobile terminal. Although the tradeoffs among analog devices, low-power ASICs, DSP cores, and embedded microprocessors in handsets remain fluid, cutting-edge base stations are beginning to employ software radio architectures. And new designs for high-end mobile radio nodes such as military vehicular radios are now largely based on some type of software radio approach. The U.S. DoD has spurred on this trend through its Programmable Modular Communications System (PMCS) study and subsequent Joint Tactical Radio System (JTRS) program. Finally, the multiband multimode flexibility of software radios appears central to the goal of seamless integration of personal communications systems (PCS), land mobile and satellite mobile services (including truly nomadic computing), toward which many of us aspire.

1In this text, the conventional notion of cellular radio is extended to embrace the idea that the propagation of RF from any SDR transmitter defines an implicit RF cell. Its size and shape is determined by the physical placement of antenna(s) and the environment. Antenna height, directivity, path loss, diffraction, and multipath loss shape the cell. A multiband, multimode SDR is uniquely suited to turn such implicit cells into explicitly managed ad-hoc cellular networks.

2In fact, the continuing interplay among military and commercial software radios plays an important role in the evolution of SDR technology. For some readers, this may impart a sense that the text skips from military to commercial points of view. The merger of these market segments around common interest in open-architecture SDR platforms is an ongoing process, complete with the common interests and occasional discontinuities highlighted in this text.
IV. THE SOFTWARE RADIO FUNCTIONAL ARCHITECTURE

Technology advances have ushered in new radio capabilities that require an expansion of the essential communications functions of source coding and channel coding. The new aspects are captured in the software radio functional model.

A. The Software Radio Functional Model

Multiband technology [1], first of all, accesses more than one RF band of communications channel at once. The RF channel then is generalized to the channel set of Figure 1-2. This set includes RF channels, but radio nodes like PCS base stations and portable military radios also interconnected to fiber and cable; therefore these are also included in the channel set. The channel encoder of a multiband radio includes RF/channel access, IF processing, and modem. The RF/channel access includes wideband antennas, and the multi-element arrays of smart antennas [2]. This segment also provides multiple signal paths and RF conversion that span multiple RF bands. IF processing may include filtering, further frequency translation, space/time diversity processing, beamforming, and related functions. Multimode radios [3] generate multiple air interface waveforms (modes) defined principally in the modem, the RF channel modulator-demodulator. These waveforms may be in different bands and may span multiple bands. A software-defined personality includes RF band, channel set (e.g., control and traffic channels), air interface waveforms, and related functions.

Although many applications do not require information security (INFOSEC), there are incentives for its use. Authentication reduces fraud. Stream encryption ensures privacy. Both help ensure data integrity. Transmission security (TRANSEC) hides the fact of a communications event (e.g., by spread spectrum techniques [4]). INFOSEC is therefore included in the functional model although the function may be null for many applications.
In addition, the source coder/decoder pair now includes the data, facsimile, video, and multimedia sources essential for new services. Some sources will be physically remote from the radio node, connected via the synchronous digital hierarchy (SDH) [5], a local area network (LAN) [6], etc., through service and network support (Figure 1-2).

These functions may be implemented in multithreaded multiprocessor software orchestrated by a joint control function. Joint control ensures system stability, error recovery, timely data flow, and isochronous streaming of voice and video. As radios become more advanced, joint control becomes more complex, evolving toward autonomous selection of band, mode, and data format. Any of the functions may be singleton (e.g., single band versus multiple bands) or null, further complicating joint control. Agile beamforming supports additional users and enhances quality of service (QoS) [7]. Beamforming today requires dedicated processors, but in the future, these algorithms may time-share a DSP pool along with the Rake receiver [8] and other modem functions. Joint source and channel coding [9] also yields computationally intensive waveforms. Dynamic selection of band, mode, and diversity as a function of QoS [10] introduces large variations into demand, potentially causing conflicts for processing resources. Channel strapping, adaptive waveform selection, and other forms of data rate agility [11] further complicate the statistical structure of the computational demand. In addition, processing resources are lost through equipment failures [12]. Joint control integrates fault modes, personalities, and support functions on processing resources that include ASICs, FPGAs, DSPs, and general-purpose computers to yield a reliable telecommunications object [13].

In a software radio, the user can upload a variety of new air interface personalities [14]. These may modify any aspect of the air interface, including whether the waveform is hopped, spread, or otherwise constructed. The required resources (e.g., RF access, digitized bandwidth, memory, and processing capacity) must not exceed those available on the radio platform. Some mechanism for evolution support is therefore necessary to define the waveform personalities, to download them (e.g., over the air) and to ensure that each new personality is safe before being activated. The evolution-support function therefore must include a software factory. In addition, however, the evolution of the radio platform—the analog and digital hardware of the radio node—must also be supported. This may be accomplished via the design of advanced hardware modules in an integrated evolution support environment, or by the acquisition of commercial off-the-shelf (COTS) hardware modules, or both.

The block diagram of the radio functional model amounts to a partitioning of the black-box functions of the ideal software radio nodes introduced above into the specific functional components shown in Figure 1-2 and listed in Table 1-1.

Not every implementation needs all subfunctions of this functional model. Thus, one may consider the functional model to be a point of departure for
### TABLE 1-1 Function Allocation of the Software Radio Functional Model

<table>
<thead>
<tr>
<th>Functional Component</th>
<th>Allocated Functions</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Coding and Decoding</td>
<td>Audio, data, video, and fax interfaces</td>
<td>Ubiquitous algorithms (e.g., ITU [15], ETSI [16])</td>
</tr>
<tr>
<td>Service and Network Support</td>
<td>Multiplexing; setup and control; data services; internetworking</td>
<td>Wireline and Internet standards including mobility [17]</td>
</tr>
<tr>
<td>Information Security*</td>
<td>Transmission security, authentication, nonrepudiation, privacy, data integrity</td>
<td>May be null, but is increasingly essential in wireless applications [18]</td>
</tr>
<tr>
<td>Channel Coding and Decoding: Modem*</td>
<td>Baseband modem, timing recovery, equalization, channel waveforms, predistortion,</td>
<td>INFOSEC, modem, and IF interfaces are not yet well standardized</td>
</tr>
<tr>
<td></td>
<td>black-data processing</td>
<td></td>
</tr>
<tr>
<td>IF Processing*</td>
<td>Beamforming, diversity combining, characterization of all IF channels</td>
<td>Innovative channel decoding for signal and QoS enhancement</td>
</tr>
<tr>
<td>RF Access</td>
<td>Antenna, diversity, RF conversion</td>
<td>IF interfaces are not standardized</td>
</tr>
<tr>
<td>Channel Set(s)</td>
<td>Simultaneity, multiband propagation, wireline interoperability</td>
<td>Automatically employ multiple channels or modes for managed QoS</td>
</tr>
<tr>
<td>Multiple Personalities*</td>
<td>Multiband, multimode, agile services, interoperable with legacy(^3) modes</td>
<td>Multiple simultaneous personalities may cause considerable RFI(^4)</td>
</tr>
<tr>
<td>Evolution Support*</td>
<td>Define and manage personalities</td>
<td>Local or network support software factory</td>
</tr>
<tr>
<td>Joint Control*</td>
<td>Joint source/channel coding, dynamic QoS vs. load control, processing resource</td>
<td>Integrates user and network interfaces; multiuser, multiband, and</td>
</tr>
<tr>
<td></td>
<td>management</td>
<td>multimode capabilities</td>
</tr>
</tbody>
</table>

*Interfaces to these functions have historically been internal to the radio, not plug-and-play.

\(^3\)Legacy refers to modes that are deployed but may be deprecated.

\(^4\)Radio frequency interference.
the tailoring of SDR implementations. In addition, many of the items in this table may be unfamiliar to some readers. The rest of the text develops the unfamiliar concepts and provides further references to the well-known aspects and standards.

B. Functional Interfaces

After identifying the functions to be accomplished in a software radio, one must define the interface points among the functional components. Figure 1-3 identifies these interfaces. The notation “RF waveform” is shorthand for air interface. The IF waveform includes most aspects of the air interface, but the signals have been filtered and converted to an IF that facilitates processing. In addition, IF processing may include A/D and D/A conversion. Baseband waveforms are almost always digital streams (e.g., of data or vocoded voice). They may also be sampled replicas of analog signals, such as digitized FM waveforms. The modem delivers what may be called decoded channel bits (“black” bits in INFOSEC jargon) to the INFOSEC function if one is present. The modem may transform analog IF signals directly to channel bits (e.g., using a despreader ASIC). INFOSEC then transforms these protected bits into clear bits (“red” bits). These bits may be manipulated through a protocol stack in order to yield source bits or network bits. Network bits conform to a network protocol, while source bits are appropriate for a source decoder. The interface to local sources of voice, music, video, etc. includes an analog transducer. Access to remote sources is accomplished via the network interface. In addition to these signal-processing interfaces, there are control interfaces mediated by the user or network (both of which are in the source

![Figure 1-3](image-url)
set in this model). Personalities are downloaded to the radio via the software object interface. The simplest mechanism for maintaining radio software after deployment is the downloading of a complete binary image of the radio. A more flexible approach allows one to download a specific new function such as a specialized voice coder (vocoder).

These interfaces may be thought of as the “horizontal” interfaces of the software radio, since they are concatenated to form signal and control flows among sources and channels. They are further characterized in Table 1-2.

In traditional radio engineering, the definition of such interfaces facilitated the design and development of the radio. Variations in these definitions from one design team to another did not matter, provided the component suppliers and the systems integrator all agreed. The idea of plug-and-play hardware and software modules has become popular in personal computing. The wireless industry seeks to benefit from the adaptation of plug-and-play technology to the software radio. This potentially elevates any functional partitioning to the role of architecture. Plug-and-play requires industrywide agreement on architecture.

C. Architecture

Since industrywide agreement on anything can be challenging, one should begin with a definition of architecture. The Random House Unabridged Dictionary defines architecture as “a fundamental underlying design of computer hardware, software, or both [25]”. While this is an agreeable definition, it provides no prescription of what “underlying design” entails. The IEEE prescribes that architecture consists of components and interfaces. This leaves one wondering what the components and interfaces are supposed to do. The Defense Information Systems Agency is the U.S. Department of Defense (DoD) agency charged with defining architecture. That agency defines architecture in terms of profiles for communications standards [26]. In its Technical Architecture for Information Management (TAFIM), the DoD characterizes architecture by analogy to “zoning laws and building codes” by which one defines the parameters for the construction of residential and industrial buildings [27].

1. Functions, Components, and Design Rules

None of the many possible definitions of architecture suit the purposes of defining architecture for the software radio. One that best relates services, systems, technology, and economics is best suited to the software radio. Architecture is therefore defined as a comprehensive, consistent set of functions, components and design rules according to which radio communications systems may be organized, designed, constructed, deployed, operated, and evolved over time. This is not inconsistent with the other definitions. But this notion of architecture more clearly addresses partitioning for plug-and-play, and the reuse of functional components. By including functions and design rules, an architecture supports component reuse, even spanning implementations that migrate among hard-
### Table 1-2 Top-Level Component Interfaces

<table>
<thead>
<tr>
<th>Interface</th>
<th>Characteristics</th>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>Analog Stream</td>
<td>Audio, video, facsimile streams</td>
<td>Continuous, infinite dimensional; filtering constraints are imposed here</td>
</tr>
<tr>
<td>Source Bitstream</td>
<td>Coded bitstreams and packets. ADC, vocoder, text data compression [19]</td>
<td>Includes framing and data structures. Finite arithmetic precision defines a coded, Nyquist [20] or oversampled dynamic range$^5$</td>
</tr>
<tr>
<td>Clear Bitstream</td>
<td>Framed, multiplexed, forward error controlled (FEC) bitstreams and packets</td>
<td>FEC imparts algebraic properties over the Galois fields defined by these bitstreams [21]</td>
</tr>
<tr>
<td>Protected Bitstream</td>
<td>Random challenge, authentication responses; public key; enciphered bitstreams [22] and packets</td>
<td>Finite dimensional; randomized streams; complex message passing for downloads; if null, this interface reverts to clear bits</td>
</tr>
<tr>
<td>Baseband Waveform</td>
<td>Discrete time synchronous quantized sample streams (one per carrier)</td>
<td>Digital waveform properties determine fidelity of analytic representation of the signal</td>
</tr>
<tr>
<td>IF Waveform</td>
<td>Composite, digitally preemphasized waveform ready for up-conversion</td>
<td>Analog IF is continuous with infinite dimensions; digital IF may be oversampled</td>
</tr>
<tr>
<td>RF Waveform</td>
<td>Power level, shape, adjacent channel interference, etc. are controlled</td>
<td>Analog RF: channel impulse response, spatial distributions via beams and smart antennas [23]</td>
</tr>
<tr>
<td>Network Interface</td>
<td>Packaged bitstreams may require asynchronous transfer mode (ATM), SS7, or ISO protocol stack processing</td>
<td>Synchronous digital hierarchy (SDH), ATM, and/or Signaling System 7 (SS7)</td>
</tr>
<tr>
<td>Joint Control</td>
<td>Control interfaces to all hardware and software; initialization; fault-recovery</td>
<td>Loads binary images, instantiates waveforms, manipulates control parameters</td>
</tr>
<tr>
<td>Software Objects</td>
<td>Download from evolution support systems (e.g., software factory)</td>
<td>Represents binary images, applets; includes self-descriptive languages [e.g., 24]</td>
</tr>
<tr>
<td>Load/Execute</td>
<td>Software object encapsulation</td>
<td>Downloads require authentication and integrity</td>
</tr>
</tbody>
</table>

$^5$A coded dynamic range is defined by the vocoder. Nyquist–dynamic range results when an analog signal is sampled so as to meet the Nyquist criteria for bandwidth recovery of the sampled signal and has been quantized with sufficient bits of sufficient accuracy to represent the two-tone spurios-free dynamic range of the application. Oversampling above the Nyquist rate can yield additional dynamic range through processing gain—see Chapter 9.
ware and software. A useful architecture partitions functions and components such that (a) functions are assigned to components clearly and (b) physical interfaces among components correspond to logical interfaces among functions. The design rules must ensure that when the hardware and software components are mated, the resulting entity accomplishes the intended functions within the performance bounds established by regulatory bodies, service providers, and users. Accommodating such diverse needs leads to complex radio systems that must be further partitioned in order manage this complexity.

2. Plug-and-Play If an architecture supports plug-and-play, then the design rules have been crafted so that hardware and software modules from different suppliers will work together when plugged into an existing system. Hardware modules will plug-and-play if the physical interfaces and logical structure of the functions supplied by that module are compatible with the physical interfaces, allocation of functions, and other design rules of the host hardware platform. Software modules will plug-and-play if there is a comprehensive but simple interface to the host environment, and if the module offers to the environment the information that it needs in order to employ it as a resource. Software radio architecture, then, defines the partitioning of functions into groups, which may subsequently be allocated to components. It defines the design rules that are appropriate for obtaining the benefits of open architecture. These include the publication of design patterns [28, 29] and interface standards. It also includes the definition of the logical levels of abstraction necessary to simplify comprehensive interfaces by hiding irrelevant details in lower layers.

D. Levels of Abstraction

Clearly, software radio functions do not all share the same logical level of abstraction. A modem, for example, supports data movement from baseband to IF, data transformation from bits to channel symbols, timing recovery, FEC and the related functions. It is therefore not accurate to think of software radio architecture as merely a collection of functions with associated interfaces. One must also identify the levels of abstraction that naturally partition the hardware and software into radio platforms, middleware,6 and host communications services, as in Figure 1-4.

In digital radios, the radio hardware platform (radio platform) accomplished most of the radio functions in hard-wired implementations, the parameters of which could be set through a microprocessor from a simple user interface or low-speed data bus. SDR platforms embody GFLOPS of processing capacity that support hundreds of thousands of lines of code (LOC). This software is partitioned into layers as illustrated in Figure 1-4. At the Radio Infrastructure

6 Middleware is software that insulates applications from the details of the operating environment (e.g., the hardware).
level of abstraction, this code moves data among the distributed multiprocess- 
ing hardware of the radio platform. At the next level of abstraction, processes 
thus distributed cooperate to form radio applications. At the highest level of 
abstraction, applications software deliver communications services to users. 
Radio applications may incorporate elaborate air interface protocols, and may 
employ standard wireline data exchange protocols like TCP/IP, so one can 
envision a much more elaborate vertical protocol slice within this four-level 
stack.

One must then define interfaces among these levels. One approach is the 
definition of an applications programming interface (API) from one horizontal 
layer to the next. The API calls may be thought of as the vertical interfaces 
among horizontal layers. This approach has been used with reported success 
on technology pathfinders [30], and will be dealt with in some detail in this 
text. Not all APIs that have been described conform to the four layers identified 
above. These four layers, however, are conceptual anchors that help organize 
the process of evolving the software radio architecture.

One current evolutionary step, for example, is the integration of CORBA 
[31] into software radio architecture. The Object Management Group (OMG) 
has defined an Interface Definition Language (IDL) in their Common Object 
Request Broker Architecture (CORBA). CORBA [34] was developed primar- 
ily to define interfaces among software modules that were not originally de- 
dsigned to work together. IDL provides facilities for defining interfaces among 
software components through the mediation of an Object Request Broker 
(ORB). Since each new component interfaces to the ORB rather than to the N 
existing components, the process of integrating a new software component is 
greatly simplified. CORBA IDL provides a rich technology base from which 
software radio finds both COTS radio infrastructure and a flexible means for 
defining interfaces among functional components. Maximum value for soft- 
ware radios requires extending CORBA to define interfaces among functions 
implemented in hardware. This has considerable benefits in a software fac-
Horizontal interfaces among functional components and the vertical interfaces among layers of abstraction partition the software radio into a matrix of manageable components. These components may then be readily integrated to create a system with desired properties. This text derives this architecture matrix and presents methods that have been proven to ensure the most critical properties of software radios. Among these is the isochronism of the real-time signal-processing streams and the computational stability of the integrated software. Consider each of these in turn.

V. BASIC SIGNAL PROCESSING STREAMS

Consider the signal streams of the software radio as illustrated in Figure 1-5. These include a real-time isochronous channel processing stream, a near-real-time environment management stream, an on-line stream to manage radio modes, and off-line data streams that support development tools.

A. The Real-Time Channel Processing Stream

The real-time channel processing stream incorporates channel coding (the RF modem functions), INFOSEC if applicable, and radio access protocols (also
called internetworking or message processing functions). Channel processing includes discrete-time point operations such as the digital translation of a baseband signal to an IF. Discrete point operations include multiplying a discrete time-domain baseband waveform by a discrete reference carrier to yield sampled in-phase IF samples.

For baseband DSP, the time between samples is on the order of milliseconds to hundreds of microseconds. This allows plenty of time for processing between samples. In the software radio’s IF stream, however, the time between samples is on the order of tens of microseconds to hundreds of nanoseconds. Such point operations require hundreds of MIPS and/or MFLOPS to gigaflops with strictly isochronous performance. That is, sampled data values must be computationally produced and consumed within short timing windows in order to maintain the integrity of the signal representation. Input/output (I/O) data rates of this stream approach a gigabit per second per IF ADC or DAC. Although these data rates are decimated through processing, it is challenging to sustain isochronism through DSP I/O interfaces and hard real-time embedded software in this stream.

Isochronous processing therefore should be organized as a hardware pipeline with sequential functions of the stream assigned to serially interconnected processors. Subscriber channels may be organized in parallel, resulting in a multiple-instruction, multiple-data-stream (MIMD) multiprocessing architecture [32]. Processors closer to the RF may be ASICS (e.g., for digital filtering and frequency translation). An important art form in software radio design is the minimization of the hardware footprint subject to the need to accommodate as many subscribers as possible. One of the major contributions of this text is to describe a proven process for accomplishing this balancing act in a way that meets end-to-end specifications in a mathematically predictable way.

B. The Environment Management Stream

The other shaded boxes of Figure 1-5 comprise the near-real time environment management stream. This stream continuously characterizes radio environment usage in frequency, time, and space. This characterization includes channel identification and the estimation of other parameters such as channel interference levels. The details of this process may be defined by specific signaling and multiple-access schemes. For example, HF Adaptive Link Establishment (ALE) includes probes and responses that characterize several assigned channels. The data is then sent on the channel that is best for the specific subscriber location. The environment management stream typically employs block operations such as fast Fourier transforms (FFTs), wavelet transforms, and matrix multiplication for beam forming. Channel identification results are needed within 540 microseconds to 2 milliseconds for the Global System for Mobile Communications (GSM) [33]. Power levels may be updated in milliseconds. Subscriber locations may be updated relatively infrequently. The block structure of such operations is readily accommodated by a MIMD par-
C. On-line Adaptation

On-line adaptation complements the near-real time dynamics as suggested in Figure 1-5. An air interface mode is a combination of parameters that defines the QoS provided by that mode. Third-generation air interfaces offer a wide range of data rates, for example. Generally, high data rates require high signal to noise ratio (SNR\(^7\)) for a required bit error rate (BER). On-line adaptation bridges across air interface modes, in order to optimize the choice of band and mode subject to the goals and constraints of the user (and/or of the network). As modes become more elaborate, users are confronted with an increasing array of QoS versus price. The burden of choosing RF band and mode in the future will be shared among the user, the network, and the wireless appliance (e.g., PDA). Thus, on-line adaptation is an area in which one can look for increasing research interest as we transition into the complexity of third-generation (3G) wireless.

D. Off-Line Software Support (The Software Factory)

Off-line tools include systems analysis, enhanced signal processing, and rehosting of existing software to new hardware or software platforms. These allow one to define incremental service enhancements. For example, an enhanced beamformer, equalizer, and trellis decoder may be needed to increase subscriber density. These enhancements may be prototyped and linked into the channel processing stream in a demonstration facility. Such an arrangement allows one to debug the algorithm(s) and to experiment with parameter settings. One may determine the value of the new feature (in terms of improved subscriber density), as well as its cost in terms of resources impact (e.g., in processing capacity, I/O bandwidth, and time delay).

In an advanced application, a software radio does not just transmit a waveform. It characterizes the available transmission channels, probes the available propagation paths, and constructs an appropriate channel waveform. It may also electronically steer its transmit beam in the right direction, select the appropriate power level, and pick an appropriate data rate before transmitting. Again, in an advanced application, a software radio does not just receive an incoming signal. It characterizes the energy distribution in the channel and in adjacent channels, recognizes the mode of the incoming transmission, and selects the appropriate processing stream. If it has a smart antenna, it also adap-

\(^7\)The SNR may be expressed in terms of unmodulated carrier and interference (CIR), signal-to-interference plus noise (SINR), or interference plus distortion.
INTRODUCTION AND OVERVIEW

<table>
<thead>
<tr>
<th>Time Criticality</th>
<th>Functionality</th>
<th>Front-End</th>
<th>Illustrative Tool Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>RF Generation</td>
<td>+</td>
<td>SPOX</td>
</tr>
<tr>
<td></td>
<td>Wideband Filters</td>
<td>+</td>
<td>P-SOS</td>
</tr>
<tr>
<td></td>
<td>Spectrum Characterization</td>
<td>+</td>
<td>Mercury</td>
</tr>
<tr>
<td></td>
<td>Beam forming</td>
<td>+</td>
<td>TMS320</td>
</tr>
<tr>
<td></td>
<td>Spreading/Despreading</td>
<td>+</td>
<td>Library</td>
</tr>
<tr>
<td></td>
<td>Equalization</td>
<td>-</td>
<td>SPW</td>
</tr>
<tr>
<td></td>
<td>Demodulation</td>
<td>+</td>
<td>COSSAP</td>
</tr>
<tr>
<td></td>
<td>Phase Tracking</td>
<td>+</td>
<td>BONES</td>
</tr>
<tr>
<td></td>
<td>Narrowband Filter</td>
<td>+</td>
<td>VHDL</td>
</tr>
<tr>
<td></td>
<td>Interference Suppression</td>
<td>-</td>
<td>Integrated Environments</td>
</tr>
<tr>
<td></td>
<td>Framing</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Forward Error Control</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interleaving</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Signaling &amp; Control</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Source Coding</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Power Management</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>BER/SNR Curve Generation</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Protocol Interactions</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Interference Characterization</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Network Architecture</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Hardware Synthesis</td>
<td>+</td>
<td></td>
</tr>
</tbody>
</table>

C = Criticality; A = Availability; * = key performance driver; + = important issue

Figure 1-6 Complementary views of a software factory.

tively nulls interfering signals, estimates the dynamic properties of desired-signal multipath, coherently combines desired-signal multipath, and adaptively equalizes this ensemble. It may also trellis decode the channel modulation and then corrects residual errors via forward error control (FEC) decoding to receive the signal with the lowest possible BER. Such operations require a family of software components and related tools including those illustrated in Figure 1-6.

The left side of the figure organizes software functions according to time-criticality. Hard real-time software may be delivered as the personality of an ASIC or FPGA. Reduced time criticality means the function is more compatible with true software implementations (e.g., as DSP code). The columns labeled C (criticality) and A (availability) identify challenge areas. Bit interleaving, for example, is not challenging either in terms of criticality or of availability. Interference suppression, on the other hand, is a critical performance driver. To the right are three columns of tool sets that represent the sophistication of the software factory. One may develop software-radio products of limited scope (e.g., under 40 k LOC) using the low-cost tools in the first column. As team size grows, or the mix of ASICs, FPGAs, and DSP hardware in the delivery environment becomes more complex, the investment of tens of thousands of dollars per design-seat pays off. The largest, most
complex systems benefit from the high-end tool suites costing upward of a million dollars (rightmost column).

The software radio should support incremental service enhancements via software tools in the software factory. These tools should assist in analyzing the radio environment, in defining the required enhancements, in prototyping incremental enhancements via software, and in testing the enhancements in the target radio environment (replete with noise and interference). The tools should make it easy to integrate and test the entire hardware-software system. They should also facilitate the delivery of the service enhancements via software and/or hardware updates, both via conventional OA&M processes and in real time over the air.

Software-based enhancements may be organized around managed objects, collections of data, and associated executable procedures that work together under the overall control of a network management system. These objects may be structured using ORBs to conform to related open-architecture software interface standards (e.g., CORBA). Such enhancements may then be delivered over the air to other software radio nodes. This is the pattern of the software-defined telecommunications network architectures described by NTT [35] and others [36, 37]. A well-integrated set of systems analysis, design, development, and rehosting tools leads to the creation of incremental software radio enhancements relatively quickly, with upgrades provided over the air as software-defined networks proliferate. Technology limitations that require hardware-based delivery (e.g., for vestpocket terminals) are met by mapping critical elements of the service enhancement to hardware (e.g., via VHDL). This leads to a wealth of implementation alternatives.

VI. IMPLEMENTATION ALTERNATIVES

Implementation alternatives for digital radios, SDR, and software radios may be characterized in the software-radio phase space of Figure 1-7. The phase space compares digital-access bandwidth to the flexibility of the processing platform. These are the two most critical architecture parameters of the software radio. Digital-access bandwidth is approximately half of the sampling rate of the ADC in the isochronous subscriber signal-processing path. Thus, for example, a 5 GHz conversion rate supports nominally a 2.5 GHz analog bandwidth, based on the Nyquist criterion [20]. ADCs with bandwidths of over 6 GHz exist [38], so digitizing RF is not impossible. If all the processing after the ADC were accomplished on a single general-purpose computer, one would have an ideal software radio receiver (the point marked X in the figure). Corresponding digital signal synthesis and up-conversion would yield an ideal software-radio transmitter.

Such extremely wideband ADCs consume substantial power and have a dynamic range of only about 30 dB. These limitations preclude practical implementations of the ideal. In addition, the digital filtering of the 5 giga-sample per second stream to access a given RF band such as 25 MHz of RF spectrum