

Chapter 2

Cognitive Radio Architecture

In this chapter we discuss some of the architectural constraints and boundaries within which a cognitive radio operates. We begin by listing the interfaces that a cognitive radio uses or implements. Then we present a brief overview of the cognitive architectures studied (primarily) in Artificial Intelligence. We then present an example architecture developed by the DARPA XG program and used as a reference in the IEEE P1900.5 standardization effort. Finally, we briefly discuss software defined radio as a platform for the implementation of cognitive radio.

2.1 Cognitive Radio Interfaces

A typical nowadays radio interacts with a number of external systems, including the radio user, the network (via a base station), sensors (e.g., spectrum sensor, GPS) and other resources accessible through the network, e.g., other users (social network), informational resources, like the Web resources, or more generally the Information Cloud. All these interfaces are shown in Fig. 2.1. Most of these interfaces were recognized as features of cognitive radio in Fette (2009).

Intelligent interaction with the user. In addition to the usual support for the transmission of audio signals, cognitive radio may support voice recognition and perception, e.g., recognition of queries and commands. This capability is already available in today's smartphone technology. Going farther, it could use speech recognition technology to perceive conversations, retrieve and analyze the content of conversations and give advice to the user (Mitola 2009b).

Interaction with sensors. The next step is vision and image processing. For example, a cognitive radio could use vision algorithms to understand the world around the user and detect opportunities to assist the user using this information.

Currently, the allocation and utilization of spectrum follows a “command and control” structure which is dominated by long planning cycles, assumptions of

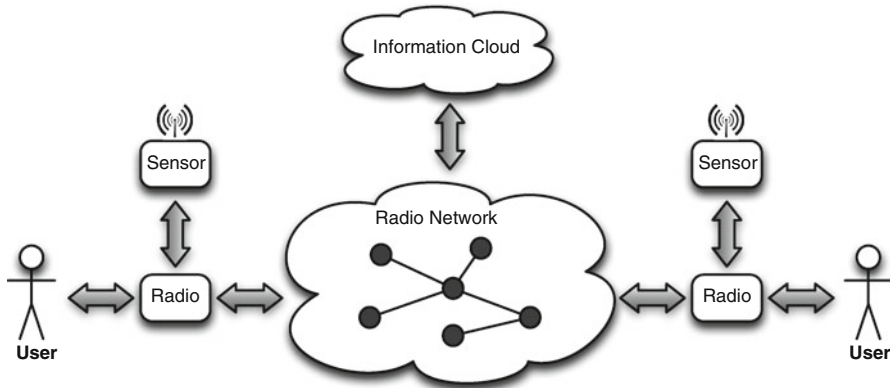


Fig. 2.1 External interfaces of cognitive radio

exclusive use, conservative worst case analysis and litigious regulatory proceedings. Using spectrum-aware radios, the management of spectrum could be transitioned into a new structure that is embedded within each individual radio. Collectively, implicitly or explicitly, the radios would cooperate to optimize the allocation of the spectrum to meet RF devices' needs (Marshall 2009). To achieve this objectives, radios would have to have the spectrum sensing capability. This could be realized by a spectrum sensor that is part of the radio.

Other sensors could also be installed on the radio. In nowadays' radios, the GPS sensor is typically used to support such applications as localization of the user's radio, mapping and direction finding applications.

Intelligent interaction with the network. Cognitive radio could provide standardized interfaces to access heterogeneous networks and support the management and optimization of network resources (Smith 2009). Referring to the OSI protocol stack, a cognitive radio's capabilities can be allocated to the particular layers:

- **Application layer:** It is an advanced PDA (personal digital assistant) with communications capability. As such, the radio would have to support simple, yet rich in functionality, interfaces for the user.
- **Network layer:** It should allow for an easy interaction with the heterogeneous networks that the radio can reach.
- **Data link and physical layers:** Cognitive radios could provide better performance in terms of connectivity, transmission capacity, reach, bandwidth, spectrum and other aspects of communication.

In this book we are focusing on the capabilities of cognitive radio that can be attributed to the Data Link and Physical layers.

Interaction with other resources on the network. Other resources available to the radio on the networks include, among others, two types of virtual networks—social networks and information networks. Social networks are groups of users who are

related via the various social relationships, e.g., *knows*, *trusts*, *manages*, and many others. The *knows* relationship is a collection of links between pairs of humans, such that one of them knows the other. Similarly, the *trusts* relationship captures the links between persons who trust other persons. Such relations can be viewed as *links*, although not as the links in the communications networks. For such links to be materialized in the communications networks, the human users must be connected to the network and the nodes to which they are connected should be linked by an active communications link.

Similarly, an *information network* can be realized as a virtual network over a communications network. The nodes in such networks are information stores, e.g., databases. The links in such networks are relations among the particular data stores, or even data items in the stores. For instance, one database may contain information about houses in a given neighborhood. Another database may contain information about people. And yet another database may contain information about the financial liabilities of particular people. These three databases are related via some relationships. For example, the people database is related to the houses database via the address where the person lives, while the liabilities database is related to the people database via the social security key. In effect, the liabilities of the persons can be traced to the financial liabilities against houses.

In today's computation and communication environments, a lot of data is stored "in the cloud", i.e., on the servers that the user even does not necessarily knows where they are. However, many users want to have a more direct control over their information resources. This issue is related to the security of information exchange. Cognitive radio might need to provide interfaces to both the user and to the cloud so that the user could have control over the whereabouts of its information resources.

2.2 Cognitive Architectures and Processes

Since cognitive radio has "cognitive" in its name, it is natural to relate the notion of cognitive radio to the cognitive architecture and processing. A cognitive architecture specifies a structure of an intelligent system (or agent). Usually, it also specifies a processing sequence for such agents. Research has been done on the topics of cognitive architecture and processing in a number of communities—cognitive science (Laird et al. 1987; Anderson et al. 2004), human factors (Endsley and Garland 2000), artificial intelligence (Langley et al. 2009), intelligent control (Albus et al. 1992; Taylor and Sayda 2005), cognitive networks (Clark et al. 2003; Mahmud 2007) and cognitive radios (Mitola 2009a). The objectives of this research were somewhat different. While the focus for cognitive science is to develop a computational model for human cognition, sometimes referred to as *the mind*, the goal for AI is mainly to build machines that just exhibit the "intelligent" behavior. For cognitive networks and cognitive radios, this goal is even more specialized—to exhibit intelligent behaviors in the technical devices, like routers, handsets and basestations.

Cognitive architectures identify blocks, like short-term and long-term memories, possibly organized into some more complex representational structures, as well as functional processing blocks. Memories contain information collected via sensing as well as agent's beliefs, goals and knowledge. Processing blocks include functions for manipulating acquired information, learning new knowledge, modifying beliefs, controlling collection of information and so on. Below, we briefly overview some of the examples of cognitive architectures known in the literature.

ACT-R (Anderson et al. 2004) is an architecture that consists of processing modules, each specialized to process a different kind of information—sensory information, beliefs, goals, actions, declarative knowledge. Associated with modules are buffers, which serve as short-term memories. Information is stored in structures called “declarative chunks”. The long-term memory contains *production rules* consisting of *condition* parts and *action* parts. Chunks include some information on their past usage; rules contain some information on their utility. Actions are executed when conditions are satisfied. Actions include both modifications of the current information structures as well as execution of some external actions, e.g., motor commands. ACT works in cycles. In each cycle it matches rules to the contents of the short-term memory and selects and executes production rules, which is guided by the goals and by the utility of the productions. The learning process can modify the production rules, both parametrically and structurally. ACT-R has been used extensively to model various aspects of human behavior.

SOAR (Laird et al. 1987) was one of the first architectures developed in the cognitive science and AI communities. The philosophy of SOAR is based on eleven hypotheses, including, among others, that intelligence must be realized with a symbol system, control is (symbolic) goal oriented, all declarative knowledge is stored in a uniform representation, long-term knowledge can be encoded by a production system, knowledge is organized (and learned) in chunks associated with the goal structures, goals can be created dynamically, weak methods (i.e., methods that arise from the interaction with the task, rather than being programmed) are part of the architecture.

The main components of the SOAR architecture are Problem Spaces (sets of states and operators that manipulate states), Long-term Production Memory (condition–action rules), Short-term Memory (context stack—objects with attributes and their values) and Preference Memory (preferences for particular elements of the context stack). SOAR's execution cycle includes two phases: elaboration (execution of all the matching productions) and decision (analysis of the preferences and putting of next problem space, operator or goal in the context stack). The cycle is repeated until the goal is reached.

Since there are so many proposed cognitive architectures, an attempt was made to develop a *schema* for instantiating particular cognitive architectures. The schema was called *CogAff* (for Cog-nition and Aff-ect) (Sloman 2001). *H-Cogaff* was a specific instance of CogAff. This architecture is presented as a grid—three columns and three layers. The layers correspond to Reactive mechanisms, Deliberative reasoning and Meta-management (or reflective processes). The columns correspond to the Perception–Central Processing–Action information processing cycle.

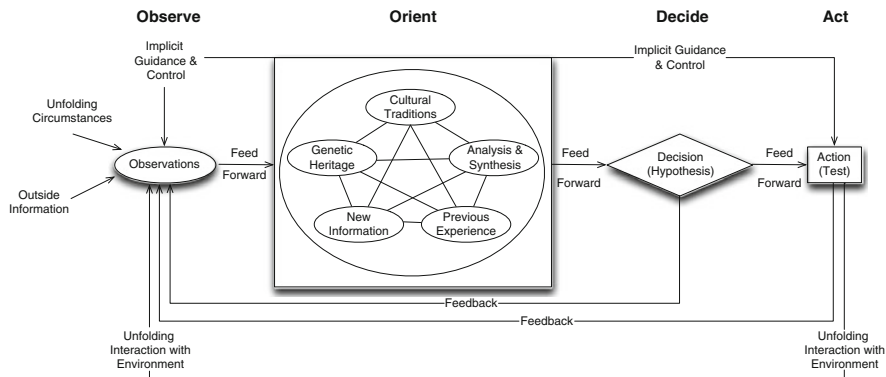


Fig. 2.2 The OODA loop

The processing cycle of CogAff abstracts what is part of many of the cognitive architectures—the three main processing steps: perception, cognition and actuation. The same pattern was advocated for intelligent controllers (cf. [Antsaklis and Passino 1992](#)), where the three processing steps were called the *perception, reasoning, action triad*. A similar idea was advocated by Boyd as a way of processing information with the goal of winning in military situations. Boyd proposed what is known as the OODA loop ([Boyd 1987](#)). The OODA loop (see Fig. 2.2) contains the four major steps: Observe, Orient, Decide and Act. Each of the steps may include other loops. Moreover, it is not a simple loop since the particular steps interact by passing both data and control information.

The Observe and Act steps loosely correspond to the Perception and Action steps in the various cognitive architectures. The mapping of the Orient and Decide steps to particular architectures would need to be analyzed on the case by case basis. It is worth noting that the Orient step includes many functions, primarily knowledge-based processing.

The OODA loop was embraced by the Information Fusion community, where it serves as a basis for situational awareness (cf. [Endsley and Garland 2000](#)). Endsley's model of cognitive situation processing includes three steps: Perception, Comprehension and Projection, where Perception corresponds to the Observe step in OODA. Comprehension and Projection correspond to Orient. Projection is the step that derives a projection of a situation into the nearest future.

2.3 Cognitive Architectures and Cognitive Radios

The use of the OODA loop for cognitive radio has been proposed by [Mitola and G. Q. Maguire \(1999\)](#). These ideas have been mentioned in many publications. Recently, [Moy \(2010\)](#) used the OODA loop to pattern the design cognitive radio upon it.

In [Amanna and Reed \(2010\)](#) the various cognitive architectures were reviewed (e.g., SOAR, ACT-R) as well as architectures of cognitive radios [e.g., CBR ([He et al. 2009](#)), Public Safety Cognitive Radio ([Le et al. 2007](#)), Open Source Cognitive Radio (OSCR) ([Stuntebeck et al. 2006](#)), DARPA xG]. However, only two of the cognitive radio solutions (architectures) were classified as being related to the OODA loop and one solution to the SOAR architecture. The general conclusion was that the developments in the area of cognitive radios is not sufficiently integrated with the developments in the domain of cognitive architectures.

The architecture of the radio developed under the DARPA xG program (cf. [Perich et al. 2010](#)), was also patterned upon the OODA loop. The cognitive capabilities of the xG radio, similarly as the most of the other cognitive radios, was limited to the functionality for achieving dynamic spectrum access.

2.4 Ontology Based Cognitive Radio

In our work we use the Ontology Based Radio (OBR) paradigm for the architecture and development of cognitive radio, introduced in 2003 ([Wang et al. 2003](#)). The main idea of OBR is flexible signaling, i.e., control messages are embedded in the payload of communication packets and the messages are expressed in terms of an ontology. In this way, control message types are not limited to the specific protocol, but instead, can be of any type expressible in the language defined by a given ontology. Unlike other radios that utilize ontologies, e.g., [Perich et al. \(2010\)](#), OBRs are not limited to a specific Application Programming Interface (API), but instead, are based on an API bound to a specific language, e.g., Web Ontology Language (OWL) ([Schreiber and Dean 2004](#)). Also, OBRs are self-aware due to the use of the feature of *reflection*, e.g., Java reflection ([Wang et al. 2004](#)). The OBR approach will be described in more detail later in this book.

The internal processing in OBR is based on the OODA loop. The “observe” part of the loop includes not only spectrum measurements, but also inputs from other sensors and from other sources of information, as shown in [Fig. 2.1](#). Moreover, OBRs can actively seek information by sending queries for specific types of information to other radios and even requesting other nodes to collect some information on demand.

The behaviors of OBRs, like most ontology-based radios, is controlled by *policies*, i.e., collections of *event–condition–action* rules. Radios monitor *events* and react to the events according to the *actions* specified in the policies that match the applicability *conditions*.

Since ontologies allow for expressing very complex types of messages and since policies allow for expressing very complex behaviors, the ontologies and the policies may need to be structured and modularized in order to make the OBR approach practical and scalable. Towards this aim, a number of architectural solutions have been proposed. One of the approaches (cf. [Raymer et al. 2006](#)), was to have a *continuum* of policies, with each component policy being crafted

to a particular constituency in the communications chain. In the case when the ontologies and policies use different languages, this approach would require a continuum of inference engines, one for each ontology/policy language.

While the OBR approach does not preclude such a compartmentalization of policies, it does not, however, provide any special architectural features to support such a fine partitioning of policies. Instead, this kind of partitioning would need to be resolved with the means of a particular ontology and policy representation language. For instance, OWL supports the *import* feature, i.e., particular ontologies can be added to a given ontology via this operation.

The ontology-based radio solutions developed under the DARPA xG program (cf. [Perich et al. 2010](#)) and the reference architecture included in the IEEE P1900.5 standard ([Group 2011](#)) adopted the approach in which two separate inference engines are identified. This architecture was influenced by Marshall's partitioning of the policies and inference modules into *endogenous components* and *exogenous components* ([Marshall 2009](#)). An exogenous component executes and enforces external policies. It addresses the radio's impact on the external environment, and ensures that the behaviors of the radio satisfy the constraints imposed by external regulations and policies. For example, an exogenous component can assist the radio in avoiding spectrum interference while searching for spectrum opportunities. Conversely, an endogenous component internally optimizes the performance of the radio through selection of operating mode and other parameters.

Based on the above perspective, the basic architecture of a cognitive radio that addresses the distinction between endogenous and exogenous components can be viewed as in Fig. 2.3. The abstract architecture of a cognitive radio comprises eight components ([Denker et al. 2009](#); [Stewart 2009](#); [IEEE P1900.5 Working Group 2011](#)):

1. Sensors. In a cognitive radio that implements DSA, sensors are used to collect information from the external environment and discover available spectrum and transmission opportunities.
2. Radio Frequency (RF). The RF component is used to transmit and receive signals.
3. Radio Platform. The radio platform includes the digital signal processing and the software control. It provides interfaces to communicate with the RF, sensors, information source and sink, and the policy reasoners.
4. System Strategy Reasoner (SSR). The SSR is an *endogenous component* of the cognitive radio. It forms strategies to control the operation of the radio. The strategies reflect the spectral opportunities, the capabilities of the radio and waveform, and the needs of the network and the users.
5. Policy Conformance Reasoner (PCR). The PCR is the *exogenous component* of the cognitive radio. It executes the active policy set to ensure that the radio transmission conforms to the policy.
6. Policy Enforcer (PE). The PE acts as a gate keeper between the SSR and the Radio Platform. It ensures that all the transmission decisions sent from SSR to the Radio Platform comply with the active policy.

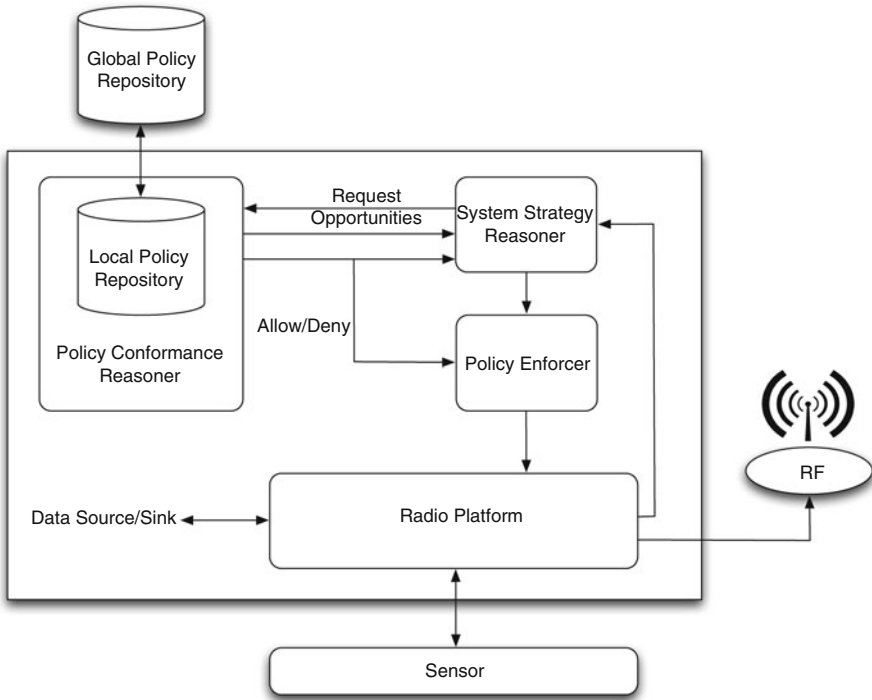


Fig. 2.3 Architecture of cognitive radio

7. **Global Policy Repository.** The Global Policy Repository stores all the policies and specific subsets configured for specific networks. The Global Policy Repository is shared across the network.
8. **Local Policy Repository.** The Local Policy Repository is within the SSR. It can download the policies from the Global Policy Repository through an interface. A radio node can store multiple sets of policies, but only one set of policies is active at any time.

In this book we focus on the functionality and the use of the SSR since this is the component that is actively involved in the process of adaptation of radio's communication parameters. Below we describe the interactions of the SSR with the other architectural components of a cognitive radio that is implemented according to the architecture shown in Fig. 2.3.

As was mentioned earlier, the OBR repetitively executes the OODA loop. The packages that carry either voice or data are processed by the Radio Platform. Symbolically, in Fig. 2.3, this is shown as the path to the Data/Source Sink. Our real interest is in the control messages—the signaling. Those messages are extracted by the Radio Platform and sent to the System Strategy Reasoner (SSR). The SSR executes the “Orient” and “Decide” steps of the OODA loop. When the SSR infers

that a packet needs to be sent to the Radio Platform (and then to the RF component), it may need to send a query to the Policy Conformance Reasoner (PCR) and ask whether the transmission is allowed or not.

The interaction between the SSR and the PCR is implementation dependent. Here we describe how this kind of interaction was specified in [Stewart \(2009\)](#) and [Denker et al. \(2009\)](#). According to these descriptions, PCR's decisions are called *transmission opportunities* that are valid for a period of time. The SSR sends a query/request to the PCR when the radio needs to change its transmission strategy, e.g., after the validity time period for a permitted transmission opportunity expires. The SSR may send one of at least two types of transmission request.

- *Unbounded transmission request.* The SSR asks the PCR to assist in identifying transmission parameters that are policy compliant. The request may not have values specified for all transmission parameters. For example, the SSR may ask: “I want to send a packet to Radio_B at time_T and at place_P—which waveform should I use?” The PCR identifies the transmission parameters that meet both the needs of the SSR's request and comply with the active policy set. Then, the PCR sends a reply back to the SSE. The reply includes the transmission parameters such as transmission power, frequency, data rate, modulation, and so on.
- *Bounded transmission request.* The SSR sends a fully bounded transmission request to the PCR, i.e., all the values of the transmission parameters are explicitly specified. The PCR evaluates the request to assess whether it complies with the active policy set and passes the result to both the Policy Enforcer (PE) and the SSR. The result can one of three types: (1) the transmission request is allowed; (2) the transmission is not allowed; (3) the transmission is allowed if specified additional constraints are added. The constraints may be acceptable values of the underspecified request parameters.

All the outgoing control messages generated by the SSR are passed to the Policy Enforcer which then ensures that all the transmissions conform to the policy. The Policy Enforcer forwards the control message to the Radio Platform. The outgoing data message and control message are merged in the Radio Platform, and then sent out through the RF.

The SSR can also send control messages to the Sensor. The sensor then collects the information of the environment and discovers which parts of the spectrum are occupied or not, accordingly.

2.5 Cognitive Radio Platform: Software-Defined Radio

In this section we provide a brief overview of software defined radio (SDR)—the platform for the implementation of cognitive radio. The definition of SDR is given by IEEE SCC 41-P1900.1 as the “radio in which some or all the physical layer functions are software defined” ([DYSPAN P1900.1 Working Group 2008](#)). The properties defined by software include carrier frequency, signal bandwidth,

modulation, network access, cryptography, channel coding (e.g., forward error correction coding) and source coding (voice, video and data). SDR is a general-purpose platform that can adapt to a wide range of waveforms, applications and products. Different kinds of waveforms at different frequencies can be implemented on the same SDR processor. Thus SDR is cost effective, versatile and easy to upgrade (reduced development cycle time) (Fette 2009).

Typically, an SDR consists of a stack of hardware and software functions, each with open standard interfaces. The SDR hardware architecture usually consists of the RF Front End, A/D converter, and the Digital Back End. First, the RF Front End amplifies the received signal, and then converts the carrier frequency of the signal to a low intermediate frequency. Second, the A/D converter converts the analog signal to a digital signal proportional to the magnitude of the analog signal. Third, the digital signal is further processed by a digital signal processor (in the Digital Back End) to perform the modem (modulation-demodulation) functions (Fette 2009).

The RF Front End usually consists of receiver and transmitter analog functions such as frequency up-converters and down-converters, filters, and amplifiers. In the full-duplex mode, the RF Front End provides some filtering to prevent the high-power transmitted signal from interfering with the lower-power received signal (Robert 2009).

The Digital Back End consists of General-Purpose Processors (GPP), Field-Programmable Gate Arrays (FPGAs) and Digital Signal Processors (DSP). A GPP usually performs the user applications and high-level communications protocols, whereas a DSP is more efficient in terms of signal processing but is less capable to process high-level communications protocols. The FPGA complements DSPs in that it provides timing logic to synthesize clocks, baud rate, chip rate, time slot and frame timing, resulting in a more compact waveform implementation. In general, the SDR hardware design is a mixture of GPPs, FPGAs and DSPs to provide a flexible platform to implement various waveforms and applications. Dedicated-purpose Application-Specific Integrated Circuits (ASIC) are not particularly suitable for SDR hardware due to their lack of flexibility (Fette 2009).

The Digital Back End is used to implement functions such as modem, Forward Error Correction (FEC), Medium Access Control (MAC) and user applications. The modem converts symbols to bits by a sequence of operations. First, the digital down-converter (DDC) converts the digitized real signal centered at an intermediate frequency to a baseband complex signal at a lower sampling rate. Second, the signal is filtered to the desired bandwidth. Next, the signal is time-aligned, despread and re-filtered. Then, a symbol detector is used to time-align signal to symbols. An equalizer is also used to correct for channel multipath effect and filter delay distortions. Finally, the symbols are mapped to bits using the modulation alphabet. Due to interference, the signal may be received with errors. FEC uses the redundancy introduced in the channel coding process to detect and correct the errors. FEC can be integrated with the demodulator or the MAC processing. After the MAC layer processing and network layer processing, the data are passed to the application layer that performs user functions and interfaces such as speaker/microphone, GUI, and other human-computer interfaces. The user

application layer usually includes vocoder, video coder, data coder and web browser functions. Typically, voice applications are implemented in DSP. Video applications are usually implemented on special-purpose processors due to the extensive cross-correlation required to calculate the motion vectors of the video image objects. Text and web browsing usually run on GPP ([Robert 2009](#)).

On top of the hardware, several layers of software are installed, including the operating system, boot loader, board support package and the Hardware Abstraction Layer (HAL). It is essential to present a set of highly standardized interfaces between the hardware platform and the software, and between the software modules so that the waveform and applications can be installed, used and replaced flexibly to achieve the user's goals ([Fette 2009](#)).

There are two open SDR architectures—Software Communication Architecture (SCA) and GNU radio ([Robert 2009](#)). SCA is a standardized software architecture sponsored by the Joint Program Office (JPO) of the US Department of Defense (DoD) for secure signal-processing applications on heterogeneous, distributed hardware. It is a core framework to provide the infrastructure to create, install and manage various waveforms, as well as to control and manage the hardware. In addition, it provides a set of standardized interfaces to enable the interaction with external services ([US Joint Program Executive Office 2011](#); [Snyder et al. 2011](#); [Communications Research Centre Canada 2011](#)).

GNU radio is a Python-based architecture that provides a collection of signal processing components to build and deploy SDR systems. It is designed to run on general-purpose computers on the Linux operating system ([Blossom 2004](#)).

Flexible Adaptation in Cognitive Radios

Li, S.; Kokar, M.

2013, XX, 156 p.,

ISBN: 978-1-4614-0968-7