# Cognitive Radio Network

Documentation V-1.0

Cognitive Radio

**Theory**

Understand the Theory behind the SCENE

Implementation

**GNU_Radio**

Tools: C, Python and XML

Testing and Validation

**Cognitive Engine**

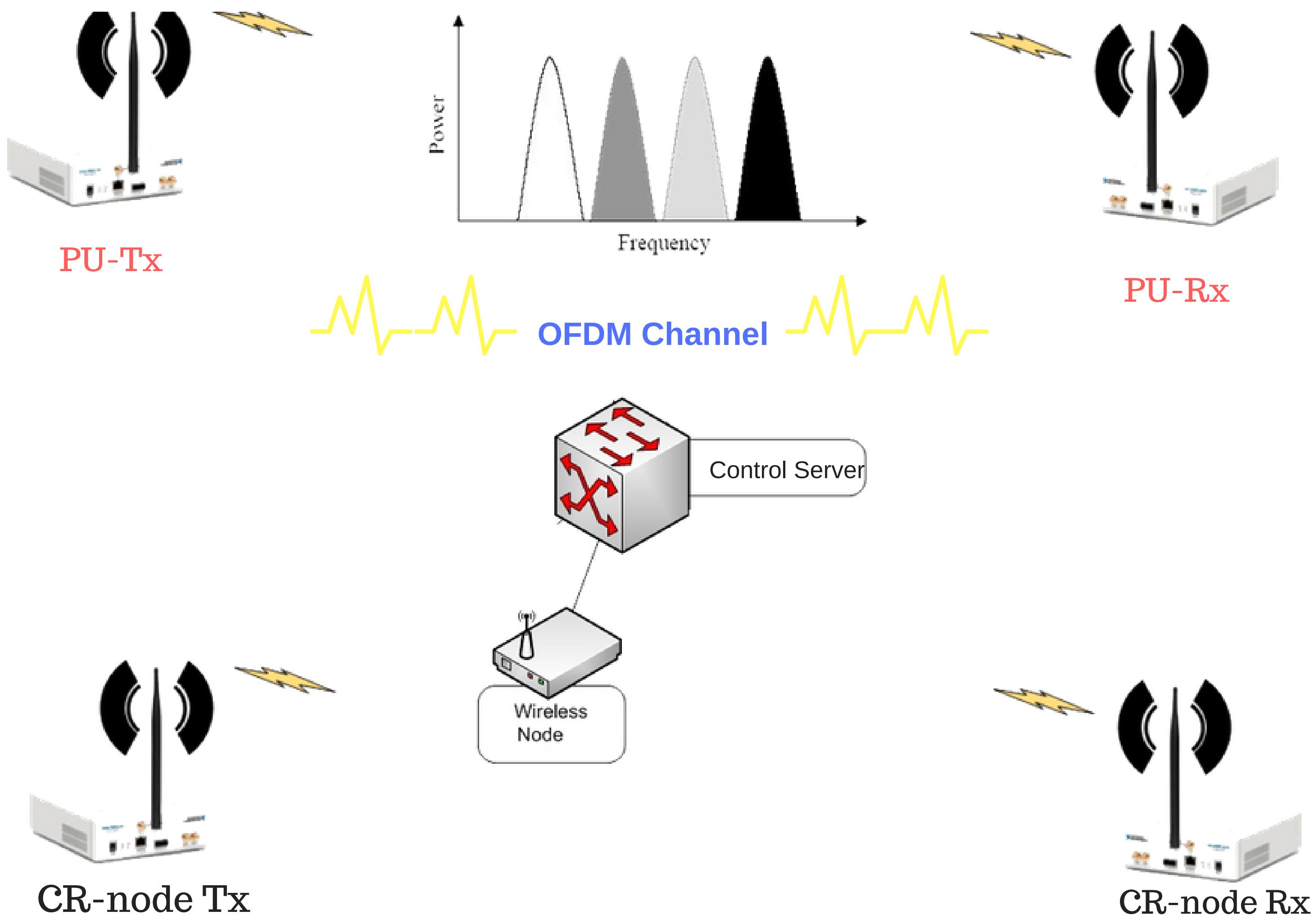
Integration

# Cognitive Radio Network

Proposal /Suggested
Scenario

# Expected Scenario

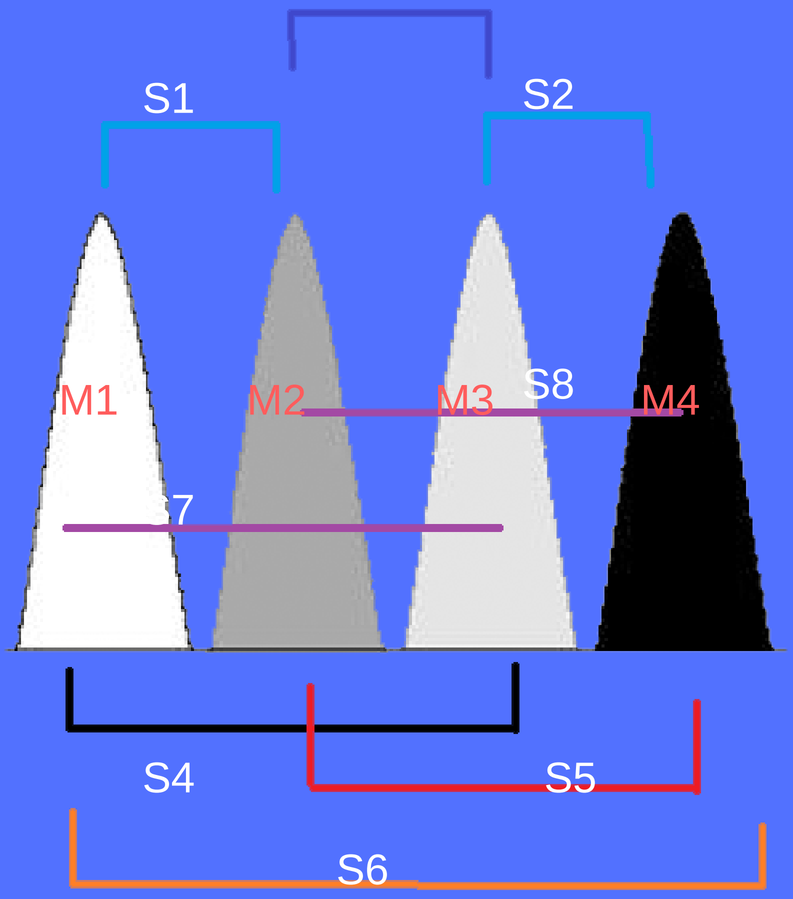
## Based on GNURadio Platform



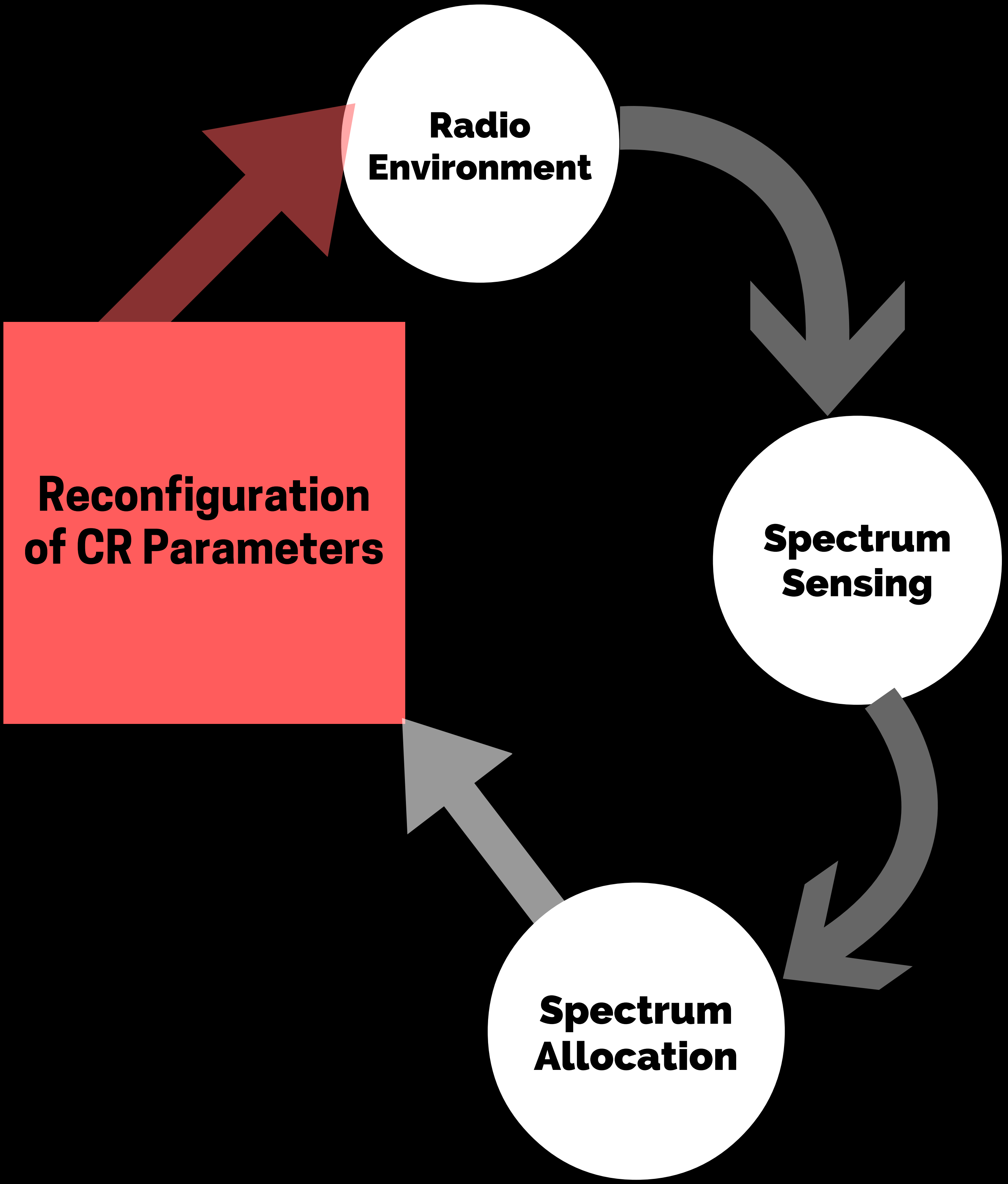
## Cognitive Control Server

Cognitive Control Server is responsible for hosting the cognitive engine which includes the spectrum sensing/monitoring Algorithm, the learning and testing function which is based on ANN-Artificial Neural Network-, the Database to store those knowledge  and Finally case based decision making.

***So, CCS-Cognitive Control Server Carries the Brain Functions***



Assume We have 4 channels,
So we have 2 to the power 4 probability of the Primary
user existence
[We assign those probability a state ID]
So the CCS determine the available state and Allocate
it to the Secondary user to exploit it while its unused.
and for sure, Its state has a different modulation scheme
data rate, tx_power,..etc.
all are based on channel statues

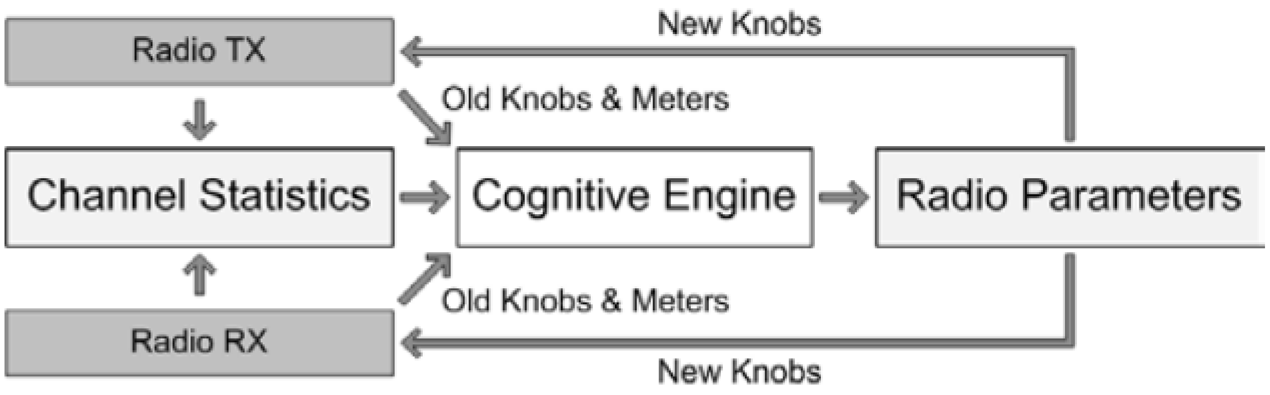


**Simple Model of Cognitive Cycle**

Observing the Outside World

Simple model of a cognitive radio where the cognitive engine interacts with the radio through its knobs and meters.

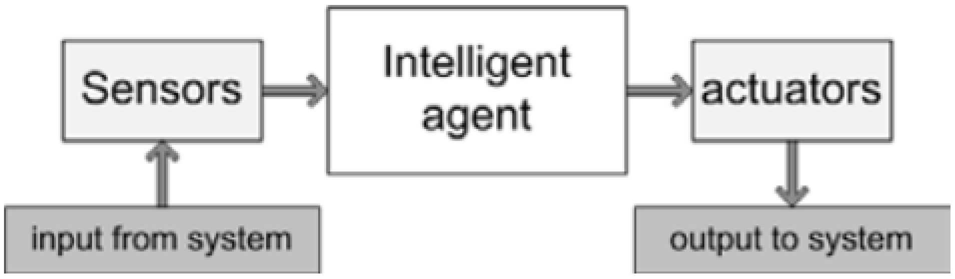
**The interaction between the cognitive engine and the radio through its knobs and meters.**
**In cognitive radio terms, the waveform is the wireless signal transmitted that represents the current settings of all of the radio's knobs.**
**Meters represent the metrics used in the radio optimization. Knobs include the type of modulation and modulation parameters, frequency channel, symbol rate, and channel and source coding. Meters include bit error rate (BER), frame error rate (FER), signal power, battery life, and computational resources.**



Briefly:
cognitive radio uses the meters to build an understanding of the environment so as to adjust the knobs to improve the communications.

## Cognitive Radio Definitions, Standards and Regulations

CR Definition: There are many definitions, standards and regulations of cognitive radio so the definition of cognitive radio has been under debate since there is no full existing CR till the moment where all the efforts are dedicated to the research field. We will handle some of CR Standards, and regulations.
and lets go back to the Definition
CR are based on building a flexible, reconfigurable radio that is guided by intelligent processing to sense its surroundings, learn from experience and knowledge, and adapt the communications system to improve the use of radio resources and provide desired quality of service.



From the shown figure.
In this way, a cognitive radio follows traditional artificial intelligence systems
These systems, act as agents that take input through sensors and respon
to the input through actuators.
The input to these systems are the radio's meters and the actuators are the radio's knobs. The intelligent agent completes the cognitive radio by providing the learning and intelligent algorithms that understand the meters and control the knobs.

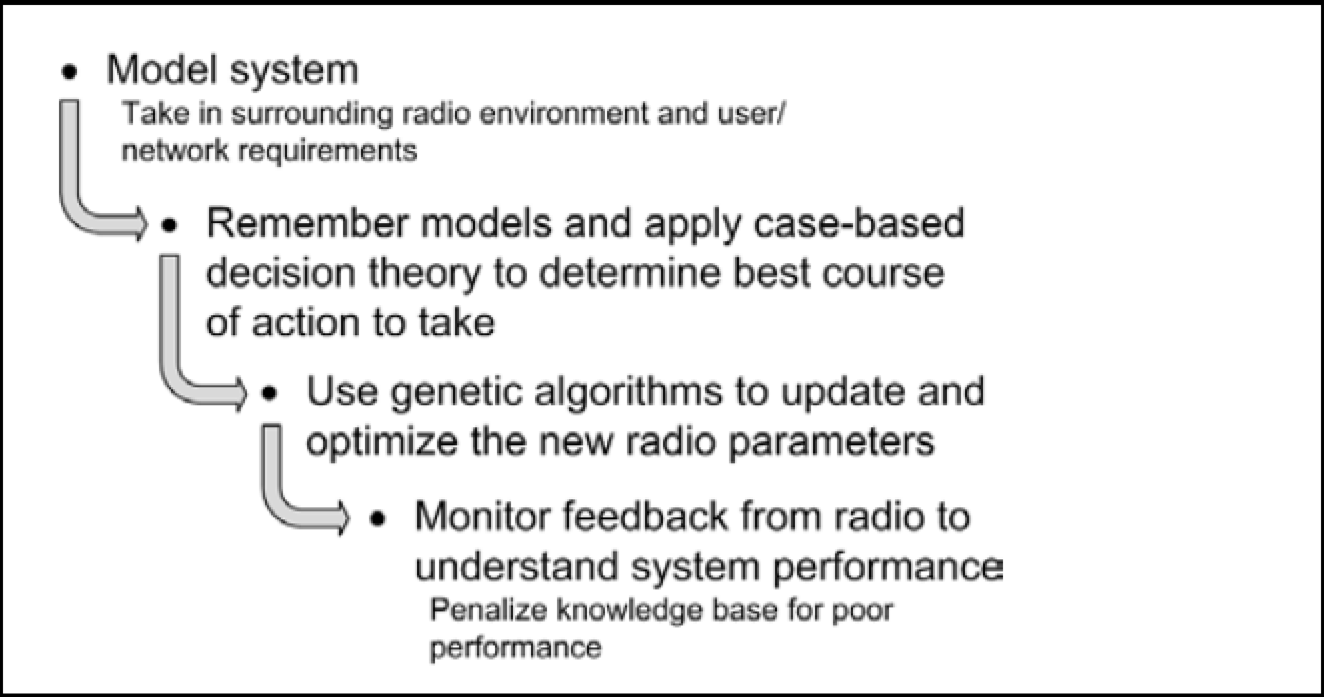
Typical artificial intelligent agent diagram that receives information through sensors and provides responses through actuators. When applied to a radio system, the agent provides the cognition to the cognitive radio.

## Cognitive Engine Design - AI

One particularly successful method to solve the multi objective cognitive radio problems is the implementation of a highly flexible Machine Learning Algorithm which are the most known branch of Artificial Intelligence.
We Will discuss The AI and Machine Learning Algorithms.
e.g: ANN, CBR and Genetic Algorithm

genetic algorithm.
The operation of the genetic algorithm optimization system easily allows updates and additions to the optimization problem space as well as the dynamic creation of chromosomes to represent the waveform, and thus, it provides a solution independent of the search space and communications system.

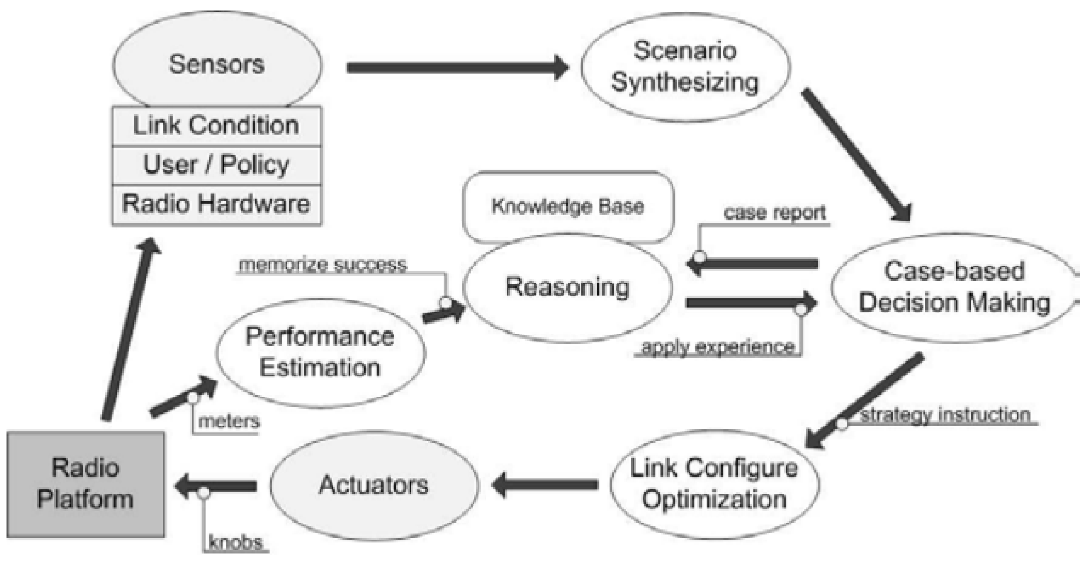
As a way to augment the optimization process, we also introduce the use of case-based decision theory. This is a memory feedback system that learns and improves the cognitive radio behavior.



• Model system
Take in surrounding radio environment and user/ network requirements
• Remember models and apply case-based decision theory to determine best course of action to take
• Use genetic algorithms to update and optimize the new radio parameters
• Monitor feedback from radio to understand system performance
Penalize knowledge base for poor performance

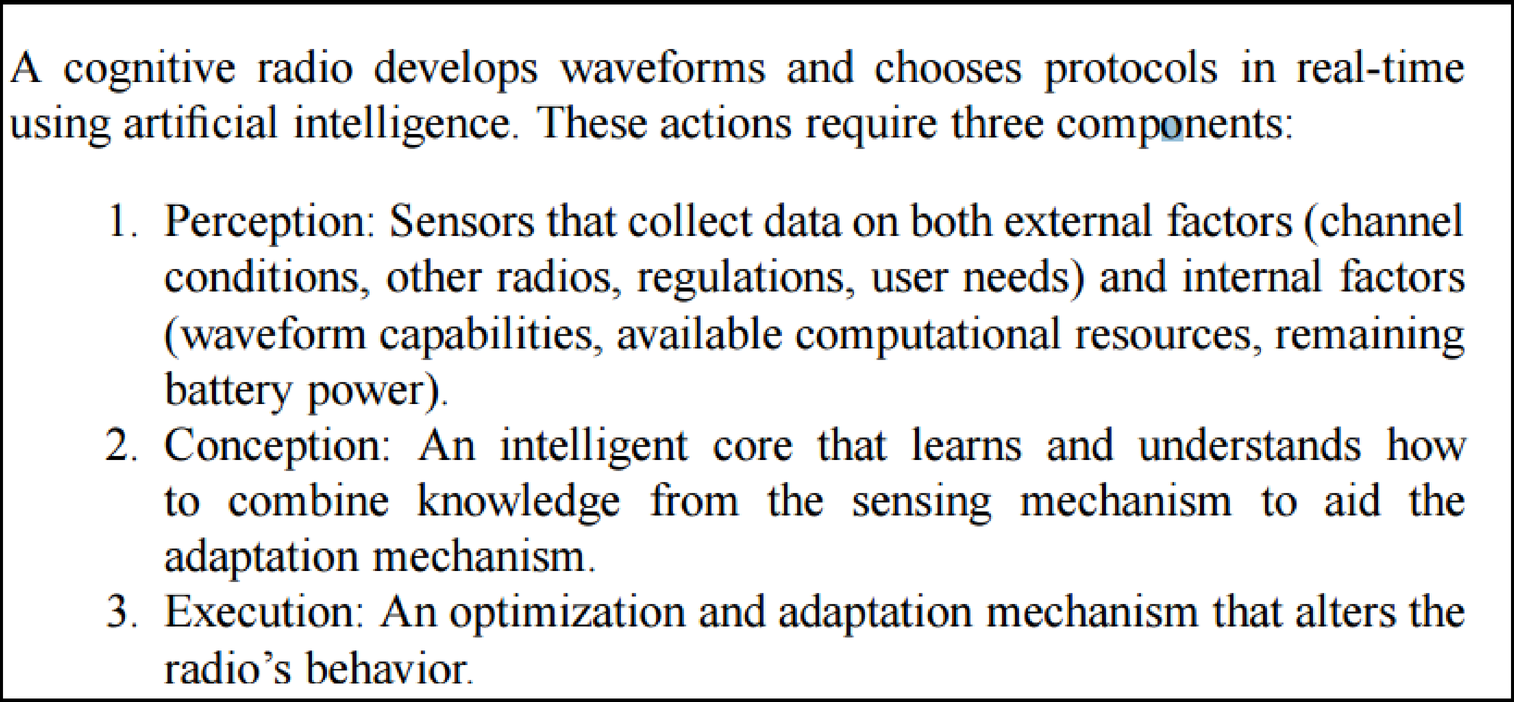
For a radio to become cognitive, we have to address many aspects in communications and computer science(multiple disciplines).
Where Its all about Cognition and Communication
where the brain power and manage the wireless network process

Cognitive Engine CE:
CE is a processing engine to translate observations about the environment to action.



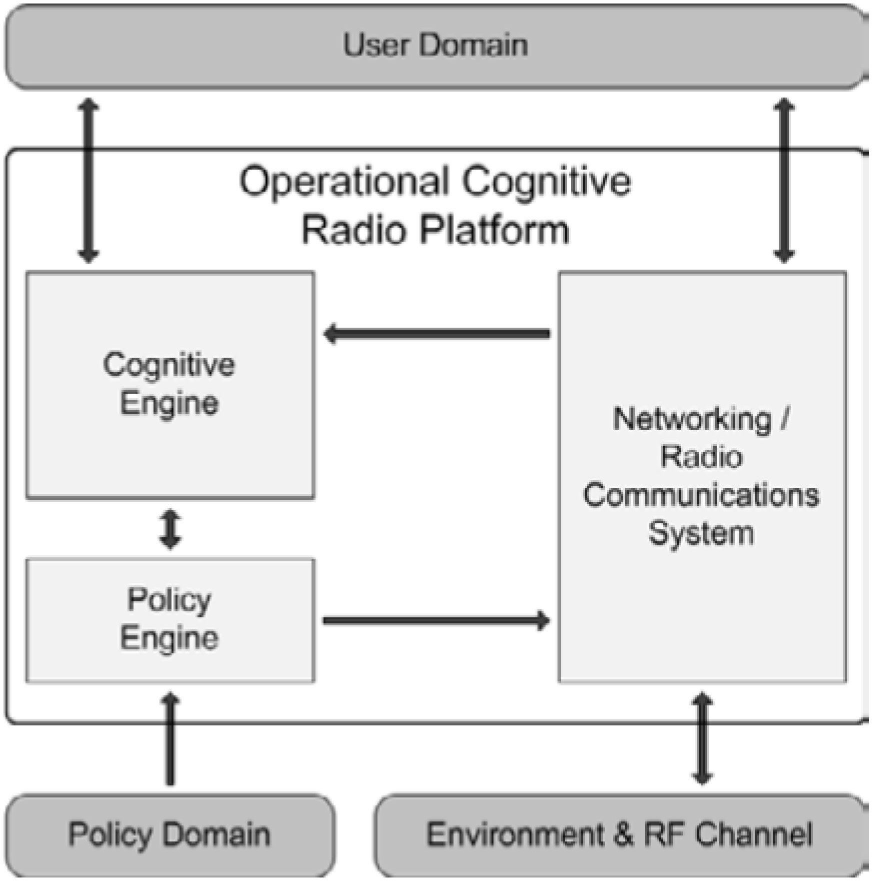
The cognition cycle with an outer loop for observation and optimization and inner loop for learning.

A cognitive radio is a flexible and intelligent radio capable of creating any waveform and using any protocol supported by the radio hardware and software.
Wave forms consist of all of the parameters that define the way
in which the radio transmits and receives information, including transmitter power, operating frequency, modulation, pulse shape, symbol rate, coding, and so forth. Protocols are the rules by which network nodes transfer information.



A cognitive radio develops waveforms and chooses protocols in real-time using artificial intelligence. These actions require three components:

1. Perception: Sensors that collect data on both external factors (channel conditions, other radios, regulations, user needs) and internal factors (waveform capabilities, available computational resources, remaining battery power).
2. Conception: An intelligent core that learns and understands how to combine knowledge from the sensing mechanism to aid the adaptation mechanism.
3. Execution: An optimization and adaptation mechanism that alters the radio's behavior.



The cognitive engine is a separate entity within that radio that works alongside the normal communications path. The engine relies on information from the user, radio, and policy domains for instructions on how to best control the communication system.

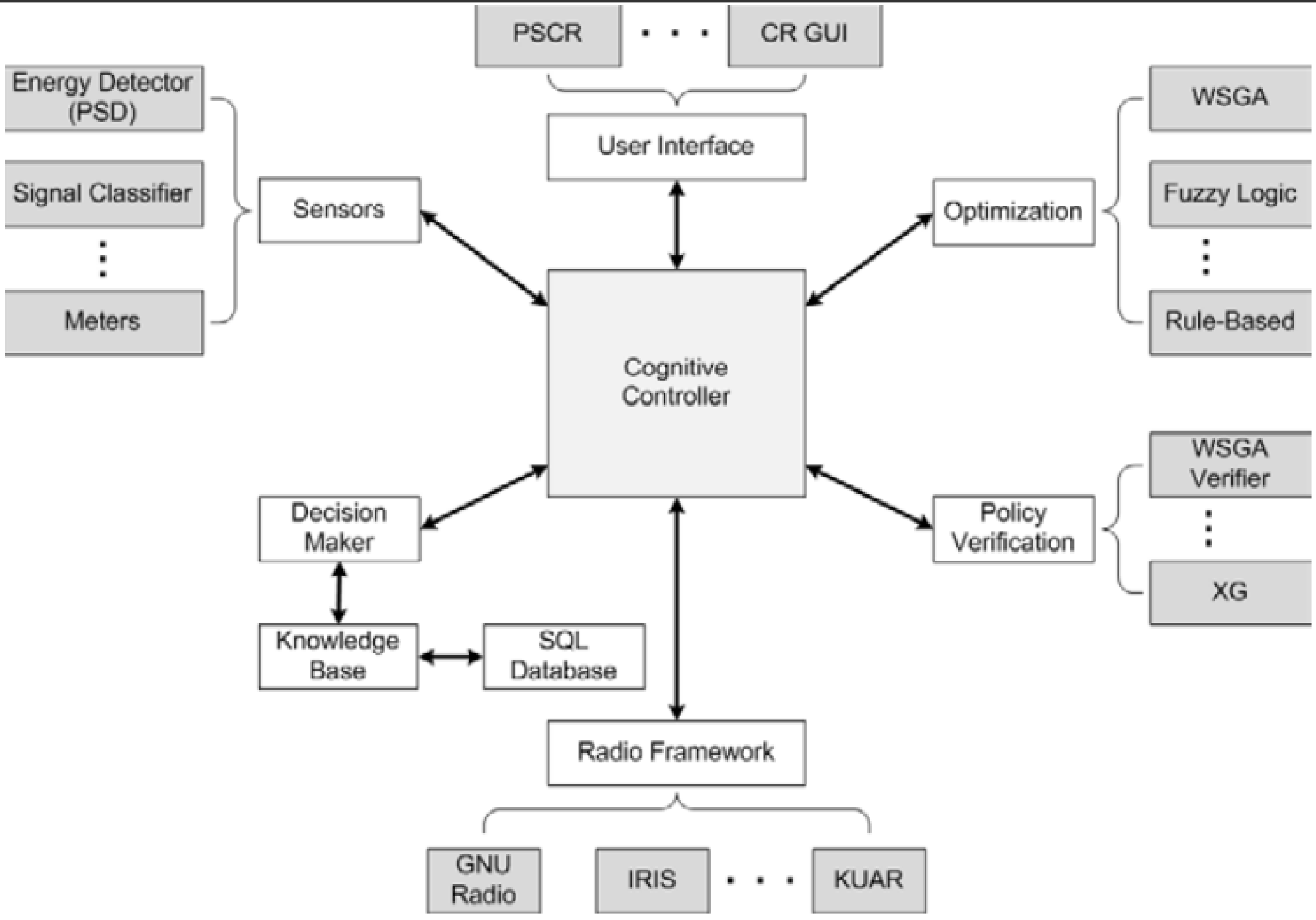
Generic cognitive radio architecture that receives input from three domains and controls a communications system.

***The user domain*** tells the cognitive engine the performance requirements of services and applications. Service and application requirements are related to the QoS measures of a communications system. As each application requires different QoS concepts like speed and latency, this domain sets the performance goals of the radio.

***The external environment and RF channel*** provide environmental context to the radio's transmission and reception behavior. Different propagation environments cause changes in the performance of waveforms that correspond to optimal receiver architectures. A heavy multipath environment requires a more complex receiver than simple line-of-sight propagation or log-normal fading. The external radio environment also plays a significant role in performance and adaptation. This environmental information helps provide optimization boundaries on the decision-making and waveform development.

Finally, ***the policy domain*** restricts the system to working within the boundaries and limitations set by the regulatory bodies as interpreted by the policy engine. The policy environment might determine a maximum amount of power a radio can use in a given spectrum or other spectrum rights with respect to other users, as was done in the 700-MHz band recently auctioned by the FCC.The rules from the FCC and other regulatory bodies impose constraints on the optimization space with respect

## Cognitive radio architecture



***cognitive controller:***
that acts as the system kernel and scheduler to handle
the input/output and timing of the other attached components.

***Sensors:***
collect radio/environmental data.

***Optimizer:***
given an objective and environment, create an optimized waveform.

***Decision Maker:***
coordinate information and decide how to optimize and
act.

***Policy Engine:***
enforce regulatory restrictions.

.

***Radio Framework:***
communicate with the radio platform to enable new waveforms and pull information from the sensors.

***User Interface:***
provide control and monitor support to the cognitive engine

# Key aspects of Design

The architecture is designed to
allow development, testing, and launching of each component separately
for low coupling between processes

## WHY
## XML

Along with a standard interface to transfer information, the system
also requires a standard for encoding the information. e.g extensible
Markup Language (XML) as a method of conveying data and information.
XML provides a method of encoding data that is open, flexible to support
new and developing sensors, and both human and machine readable. XML
also has a standardized
format and methods of verifying the format, via a document type definition
(DTD), to make integration with the cognitive controller easy. Finally, XML
is an open, simple standard with many tools available to read and write it,
including libraries for almost any programming language.

**XML is just a container format to pass the data**

# Sensors

Sensors collect data from the radio or other systems to describe and model the environment. Environmental data can include almost anything that will help the radio adjust its behavior, including radio propagation, interference models (temperature), position and location, time, and possible visual cues. In whatever manner the data is collected, the important aspect of a sensor is having a standard approach to how data is transferred to the cognitive controller.

The application programming interface (API) is described as a simple state machine with a few important states:

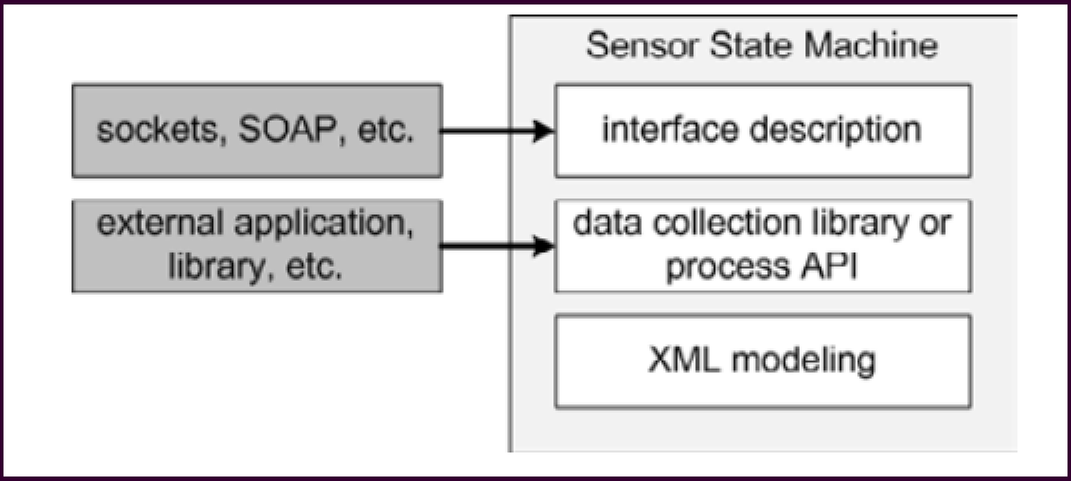
- Initialization;
- Waiting for data request from cognitive controller;
- Collecting data and building model;
- Transferring model to cognitive controller.

**Initialization**

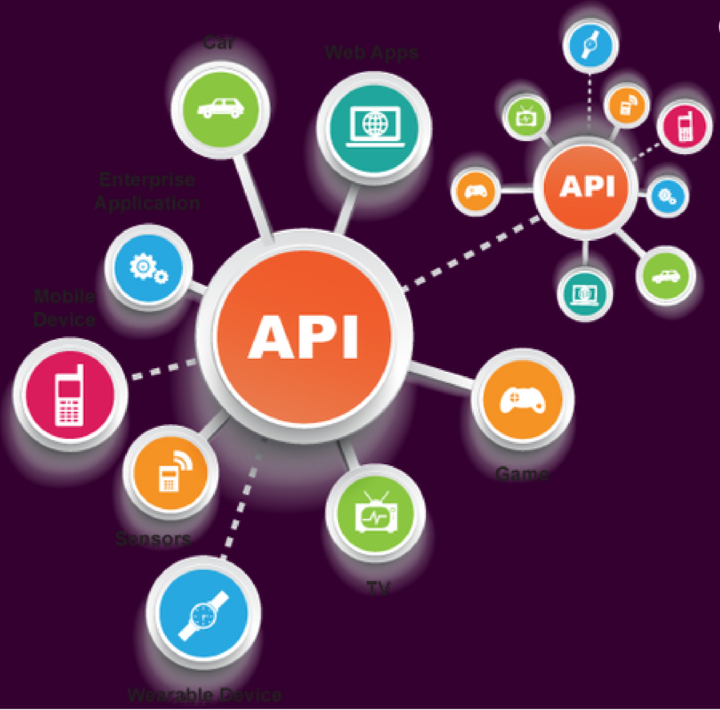
Initialization builds the proper interfacing to the cognitive controller.
The sensor then enters a wait state to listen to its interface for a request
for data from the cognitive controller. When the sensor receives a request,
it performs its data collection process, possibly by calling external libraries or applications, and then packages the data into an XML format to describe the sensor data.

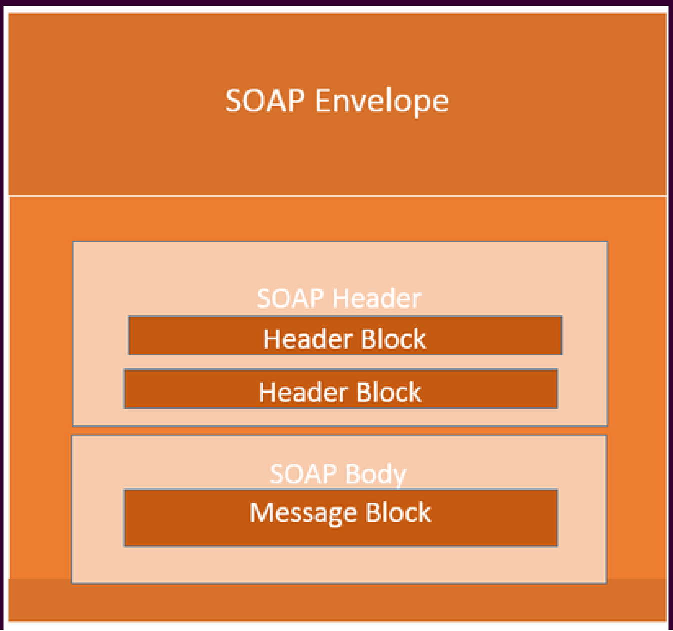
The XML data is transmitted to the cognitive controller, and the state machine returns to its wait state.

the structure of a sensor

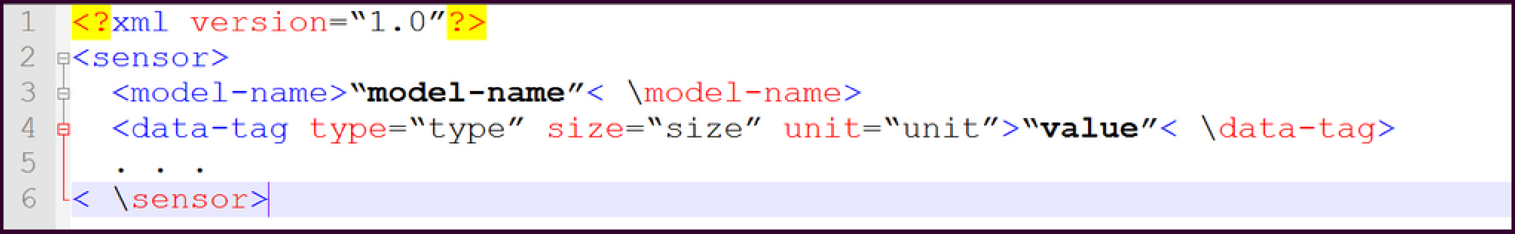

Sensor State Machine
sockets, SOAP, etc.
interface description
external application, library, etc.
data collection library or process API
XML modeling

the cognitive engine sends information to the sensor through some generic interface.
Sockets and Simple Object Access Protocol (SOAP) are two simple ways to communicate information between software programs either collocated on the same hardware or distributed and connected through a network layer.
These are only two possible methods for providing the communications between the sensor and the engine.
Functions and processing algorithms are retrieved through an externa application or library.
The use of an external interface here allows the sensor to access new different functionality through simple and independent updates to a libra or API.





lists the basics of the XML format for conveying sensor information to the cognitive controller



```
<?xml version="1.0"?>
<sensor>
  <model-name>"model-name"< \model-name>
  <data-tag type="type" size="size" unit="unit">"value"< \data-tag>
   .  .  .
< \sensor>
```

Skeleton XML format for describing a sensor

The first line simply defines this as an XML v1.0 file. The next line says that this is a model from a sensor process. The particular model name is ,then the character data of the "model-name" tag.

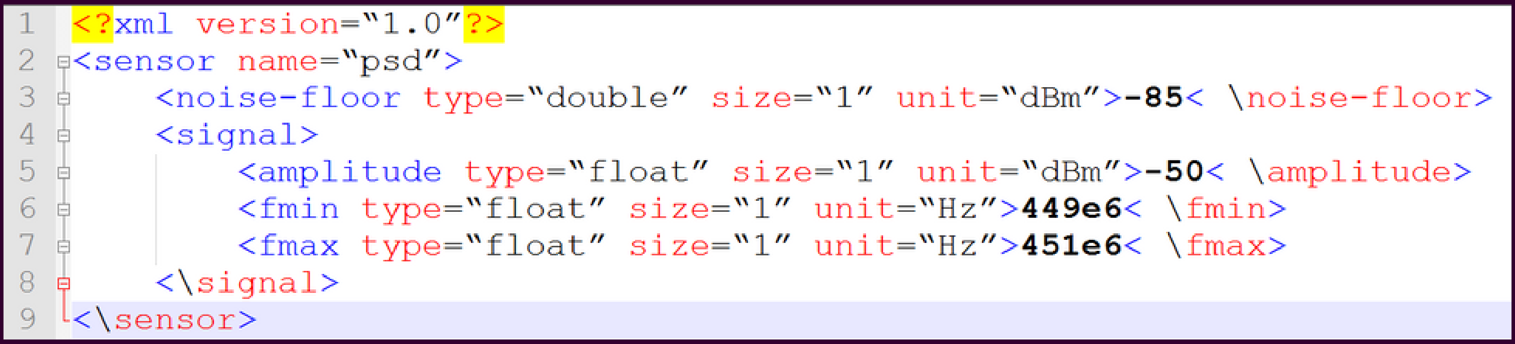
The remaining tags contain the model data. The important practice for proper representation of model data is

shown in the fourth line where the data type, size, and unit are defined. The specification of the data type can be any type used by a particular database or language specific to the processing of the data. For instance, data used with a MySQL database can have the type "int," "float," or "char." The size information indicates the number of items included in this data tag; basically, this is defined to help represent vector data. If this attribute is ignored, a value of "1" is assumed. The "size" attribute is an admittedly ugly method of enabling lists or vectors of data that are comma- or space separated values.

The final standard attribute describes the unit value the data represents, for example, dBm for power or Hz for frequency

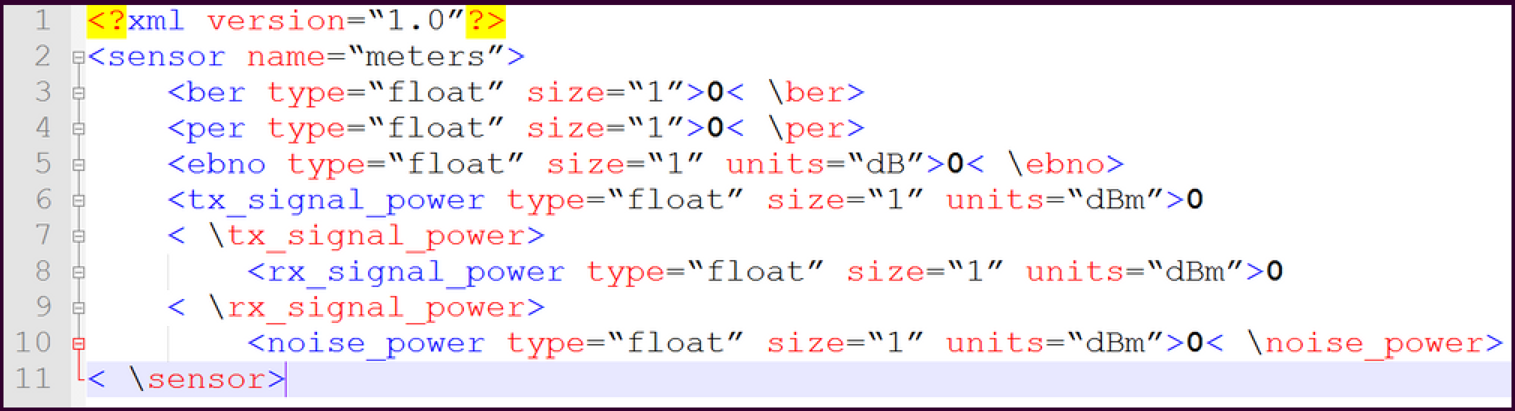
## example of a possible model received from a power spectral density (or energy detector) sensor.

This example contains information on the system’s noise floor at -85 dBm and one signal present in the environment that has a received amplitude of -50 dBm between the frequencies 449 to 451 MHz.



```
<?xml version=“1.0”?>
<sensor name=“psd”>
    <noise-floor type=“double” size=“1” unit=“dBm”>-85< \noise-floor>
    <signal>
        <amplitude type=“float” size=“1” unit=“dBm”>-50< \amplitude>
        <fmin type=“float” size=“1” unit=“Hz”>449e6< \fmin>
        <fmax type=“float” size=“1” unit=“Hz”>451e6< \fmax>
    <\signal>
<\sensor>
```

## Example XML description of a meters sensor



```
<?xml version=“1.0”?>
<sensor name=“meters”>
    <ber type=“float” size=“1”>0< \ber>
    <per type=“float” size=“1”>0< \per>
    <ebno type=“float” size=“1” units=“dB”>0< \ebno>
    <tx_signal_power type=“float” size=“1” units=“dBm”>0
    < \tx_signal_power>
        <rx_signal_power type=“float” size=“1” units=“dBm”>0
    < \rx_signal_power>
        <noise_power type=“float” size=“1” units=“dBm”>0< \noise_power>
< \sensor>
```

This sensor collects the system information such as noise power, signal power, bit error rate, battery life, or any other available meter. This information is very important to the optimization process, and lack of this information or wrong information could seriously impact the optimization process.