An Introduction to a UML Platform Independent Model of a Software Radio

ICT'2002 Tutorial
Sunday, June 23rd, 2002

Michel Barbeau
Francis Bordeleau
Jeff Smith

Objectives

- Introduce the concept of Software Radio
- Give a general overview of the Software Radio UML Model that is being standardized by the OMG
- Discuss issues related to the implementation of the UML model
- Promote Software Radios

Outline

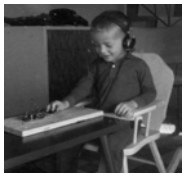
1. Introduction to Software Radio
2. Introduction to UML
3. Software Radio UML Model
4. Work in Progress

→ 1. Introduction to Software Radio
2. Introduction to UML
3. Software Radio UML Model
4. Work in Progress

Highlights of the wireless history

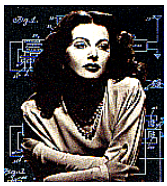
R. A. Fessenden, 1900
Radio-Telephone

M. Barbeau, 1962
Sending morse code



J. Mitola III, 1991
Software radio

G. Marconi, 1895
wireless telegraphy



H. Lamarr and G. Antheil, 1942
frequency-switching

R. Barbeau, 1962
Fixing a FM radio

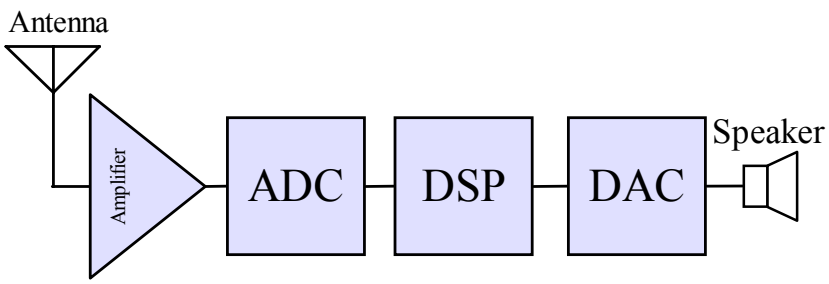
What is Digital Signal Processing?

- Measuring analog signals
- Recording measurements as series of numbers
- Processing the numbers
- Converting back to analog signals

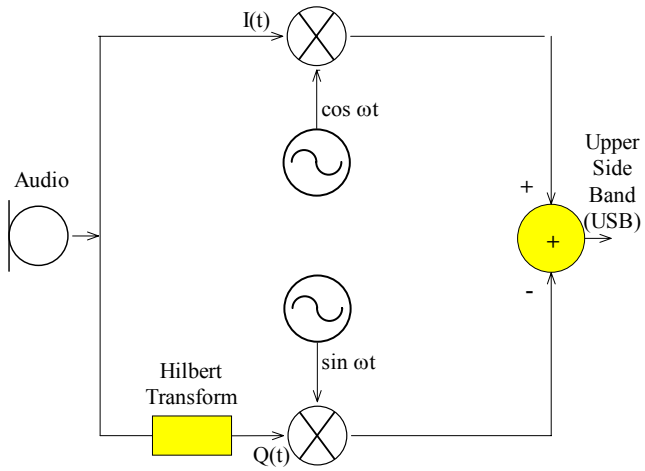
What is a Software Radio?

- Radio with functions implemented in software
- Multi-band antennas
- Transmitter
  - conversion of digital to analog (DAC), possibly to an IF and then to RF
- Receiver
  - wideband analog to digital conversion (ADC), down conversion, demodulation
- Increasing flexibility via increased programmability
- Require a multi-disciplinary approach

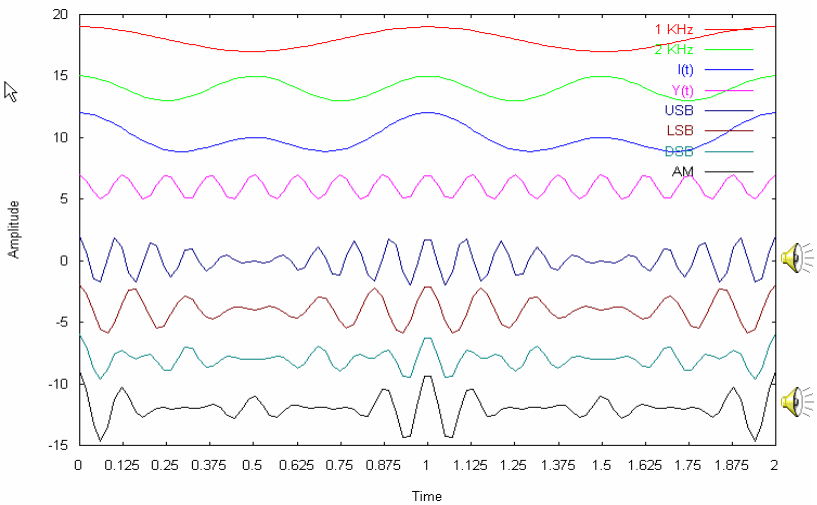
A Simple Software Radio Architecture



Half-complex mixer



Waveforms



Single Side Band (SSB) Generation

Method: phasing method

$I(t)$, Amplitude of an audio signal versus time (real signal with a two-sided spectrum)

Quadrature signal $Q(t)$

All frequency-components of $I(t)$ phase shifted by 90° (Hilbert transform)

Analytic signal

Complex signal with only positive-frequency components

$I(t) + jQ(t)$

Carrier: Complex oscillator

$Y(t) = \cos \omega_0 t + j \sin \omega_0 t$

Upper Side Band (USB) signal: Product of analytic signal with complex oscillator

$Y(t)[I(t) + jQ(t)] = I(t)\cos \omega_0 t - Q(t)\sin \omega_0 t$ Only real part is computed

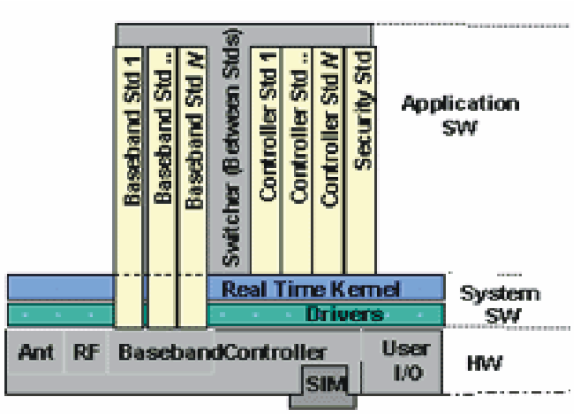
What is a Software Defined Radio?

"Software-defined radios let service providers reprogram base stations to reassign channels as standards change and the mix of analog vs. digital users shifts. And engineers envision handsets that someday will download from any network whatever code is needed to reprogram themselves to access a wireless service or run a mobile application." EE Times November 15, 2000.

Application Domains

- Military and Commercial
  - Air traffic control
  - Cellular phone
  - Computer networks
  - Pervasive computing
  - Satellite telecommunications

Software Defined Radio Architecture

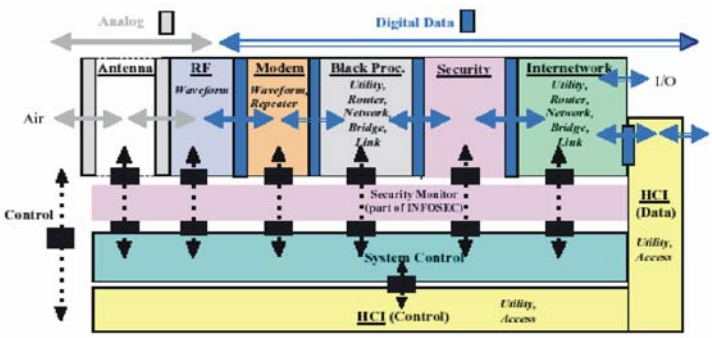


Handheld Multiple Service Model Using SDR

Current Standardization Efforts

- Software Communications Architecture (SCA)
  - **–Published by Joint Tactical Radio System (JTRS) Joint Program Office (JPO)**
  - **–Developed by the Modular Software-programmable Radio Consortium (MSRC), which is composed of Raytheon, Harris, BAE, and ITT**
- SDR Forum
  - **More than 138 members including Boeing, CRC, France Telecom, Harris, Mercury Computer Systems, Motorola, Nortel Networks, Raytheon Systems, Spectrum, Thales, etc.**
- OMG SWRadio DSIG
  - **–Develop the Platform Independent Model (PIM) and some Platform Specific Models (PSMs)**
  - **–Led by Mitre, Mercury Computer Systems, and Raytheon**
  - **–Main contributors includes MSRC members, Mercury Computer Systems and Carleton University**

Software Reference Model (SCA)



Current Radio Issues

- There is a large inventory of waveforms and communication services. This is especially true for the Armed Forces and is also a growing concern for the commercial waveform migration.
- Waveform and Service Applications are typically not portable across hardware platforms.
- Interoperability is limited based upon radios' reprogrammability.
- Many current radios are not re-programmable; they must be replaced when new capabilities are desired.
- Radios from different manufactures implementing the same waveform cannot inter-operate with each other.
- Command and Control is different for each Radio.
- General software solutions to these issues tend to increase radio size, weight and power, especially critical to mobile radios.

Scope of Proposals Sought by OMG

The scope of the RFP is to provide software radio infrastructures to:

- Yield interoperable waveforms and radios
- Yield programmable and re-programmable radios by defining APIs within the radio that waveforms are built upon
- Promote waveform software portability and reuse across hardware platforms
- Be open and documented to allow 3rd party waveform development
- Maximize independence of waveform from specific hardware solutions

Scope of Proposals Sought by OMG (cont'd)

- Be extendible to new waveforms and/or hardware components
- Providing a standard Application Program Interface (API) for code portability with different OSs.
- Updating instead of replacement when new capabilities desired.
- Storage provisions for a library to contain the waveform inventory.
- Be scaleable from low-capability (mobile phones) to high-capability radios (e.g., basestations, multi-channel radios, etc.).
- Provide for rapid technology insertion for new hardware and software technologies as they become available over time

1. Introduction to Software Radio
2. Introduction to UML
3. Software Radio UML Model
4. Work in Progress

Unified Modeling Language

• Graphical Language for { Visualizing, Specifying, Constructing, Documenting } the artifacts of Software-based systems

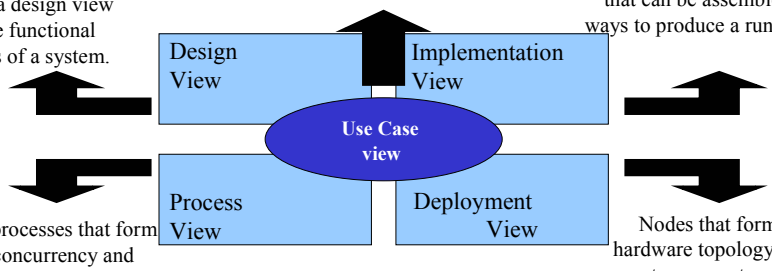
- Standard for object oriented software modeling
- Allows modeling different aspects (different views)
- Allows modeling systems at different levels of abstraction
- Scenario driven process

Different Views

Classes, interfaces, and collaboration that form the vocabulary of the problem and its solution; a design view addresses the functional requirements of a system.

Uses cases that describe the behavior of the system as seen by its end users, analysts and testers.

Components used to assemble and release the physical system; an implementation view addresses the configuration management of the system's releases, made up of somewhat independent components that can be assembled in various ways to produce a running system.

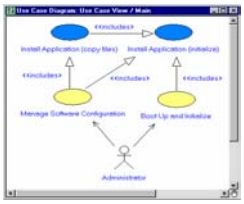


Threads and processes that form the system's concurrency and synchronization mechanisms; a process view addresses the performance, scalability, and throughput of the system.

Nodes that form the system's hardware topology on which the system executes; a deployment view addresses the distribution, delivery, and installation of the parts that make up the physical system.

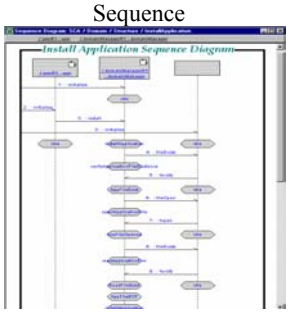
Different Diagrams

Use Case

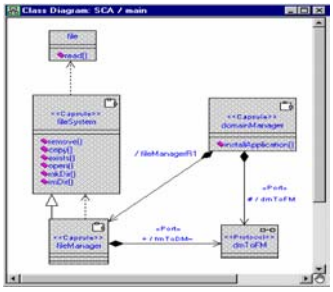


Use Case Description

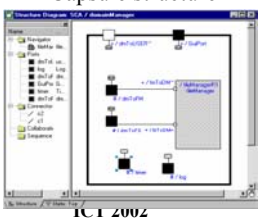
Sequence



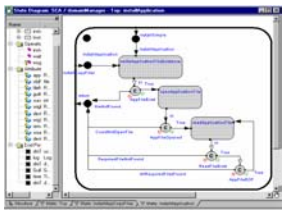
Class



Capsule structure



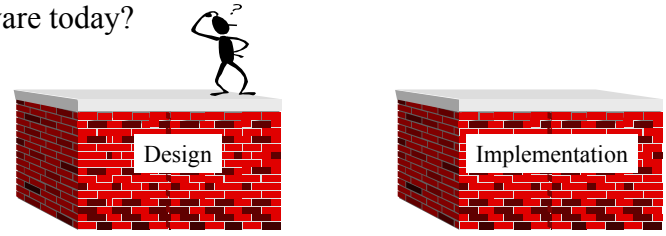
Statecharts



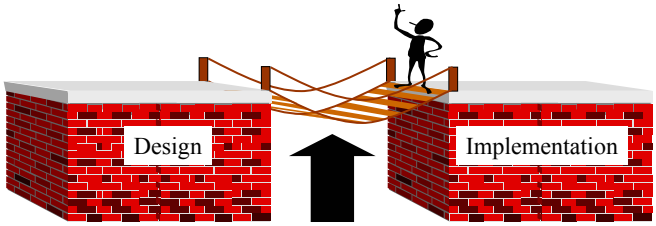
From Design to Implementation

How do we build software today?

Traditional

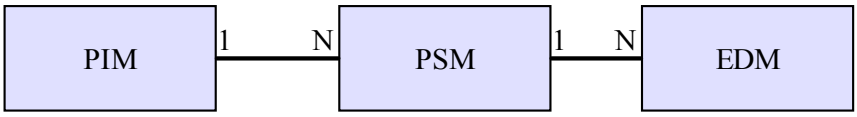


UML-RT

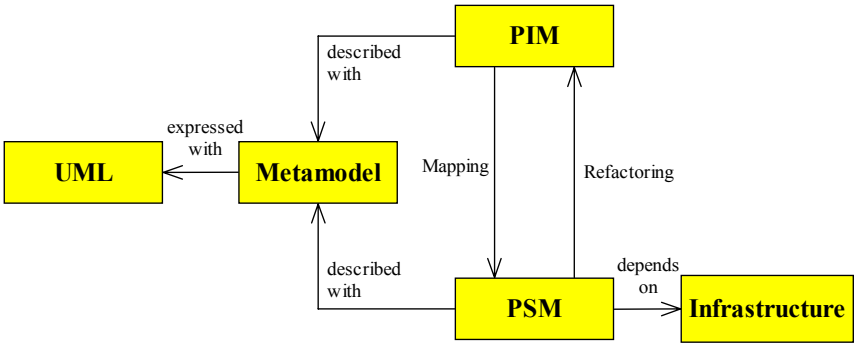


Code Generation

Model Driven Architecture



Model Driven Development



1. Introduction to Software Radio
2. Introduction to UML
3. Software Radio UML Model
4. Work in Progress

Key Concept: Waveform

- Transformations
- In a SWR transmitter, encode information and produce a signal that can be transmitted over the air
- In a SWR receiver, decode a signal received over the air and produce information

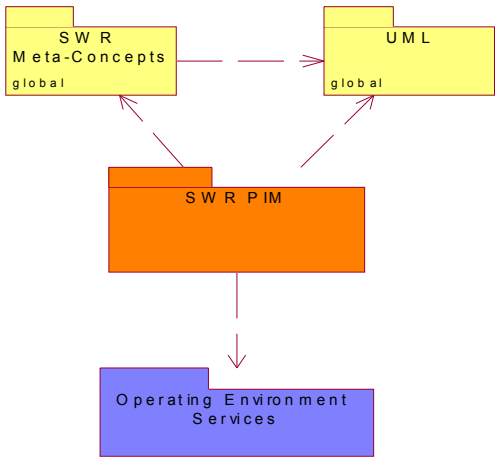
Key Concept: Application

- Is a program that realizes a given waveform
- Provides an interface through which it can configured, controlled and monitored.

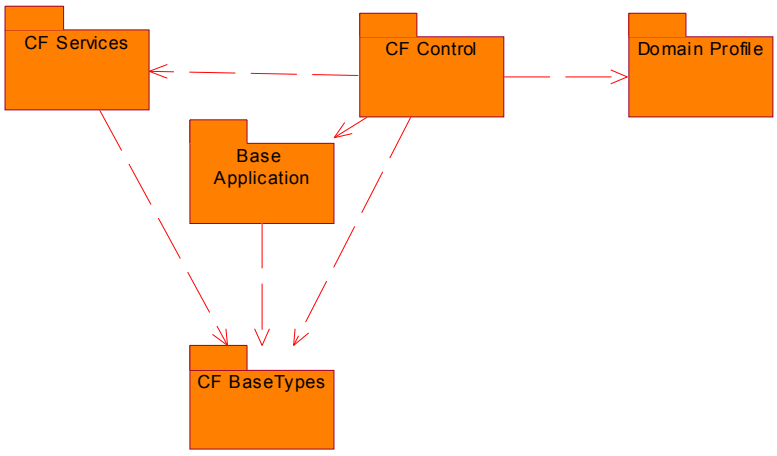
Key Concept: Domain

- Is a set of hardware devices and applications
- A given domain is under the control of a domain manager

SWR PIM Dependencies



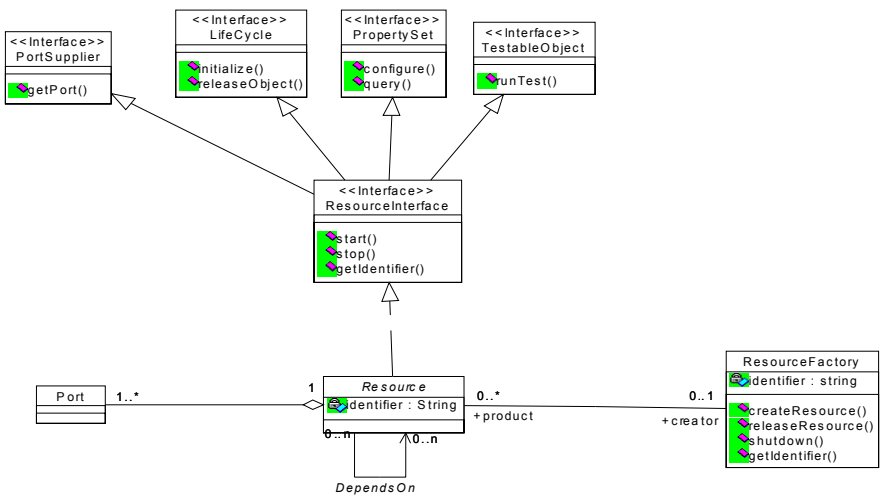
SWR PIM Package



Base Application Package

- The Base Application package defines the elements that are at the core of an application

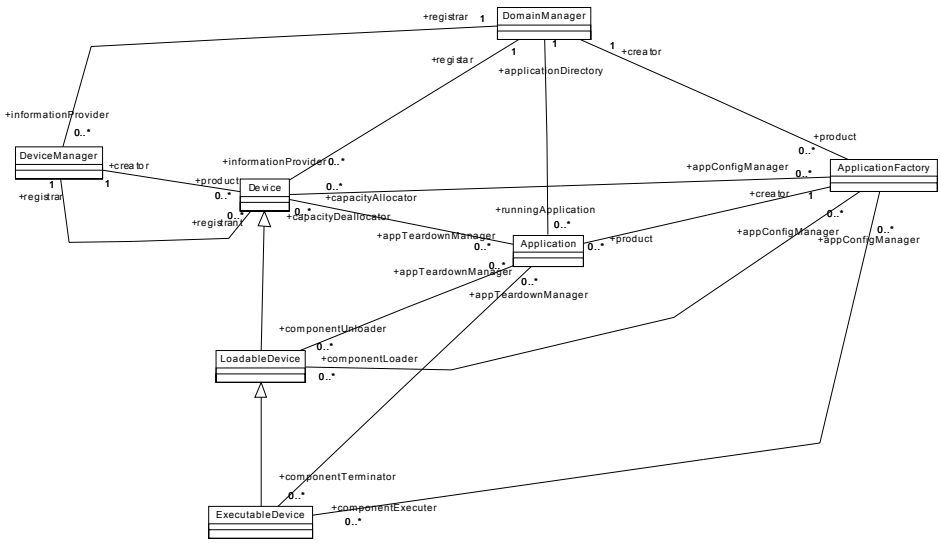
Base Application Package



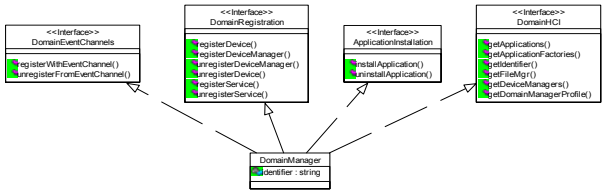
CF Control Package

- Control and management of resources and ports
- Management of a domain:
  - Applications
  - Hardware devices

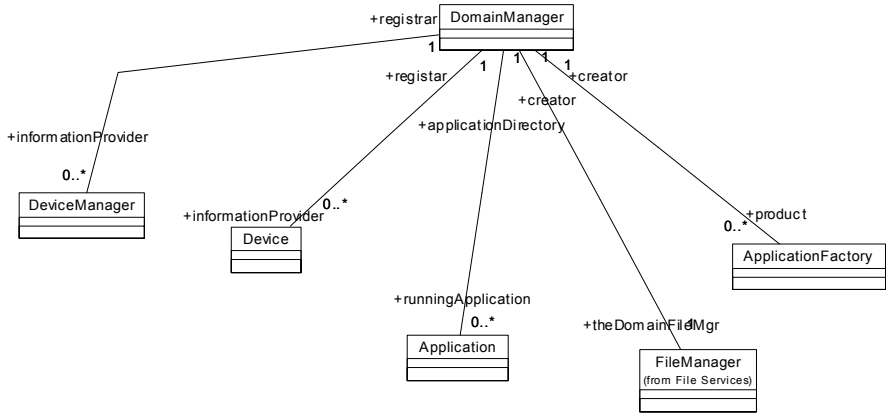
CF Control System Relationships



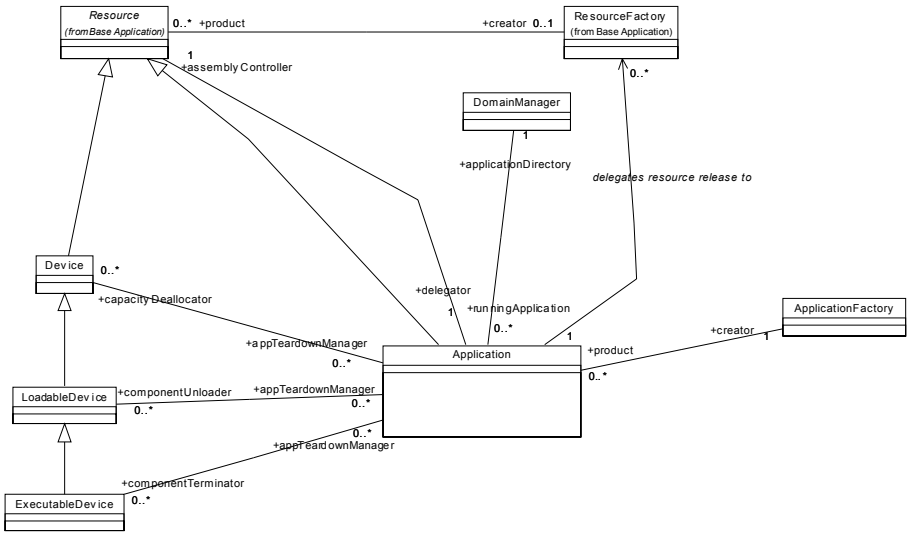
CF Control Package: Domain Manager



CF Control Package: Domain Manager



CF Control Package
Management of Applications



CF Control Package: Device

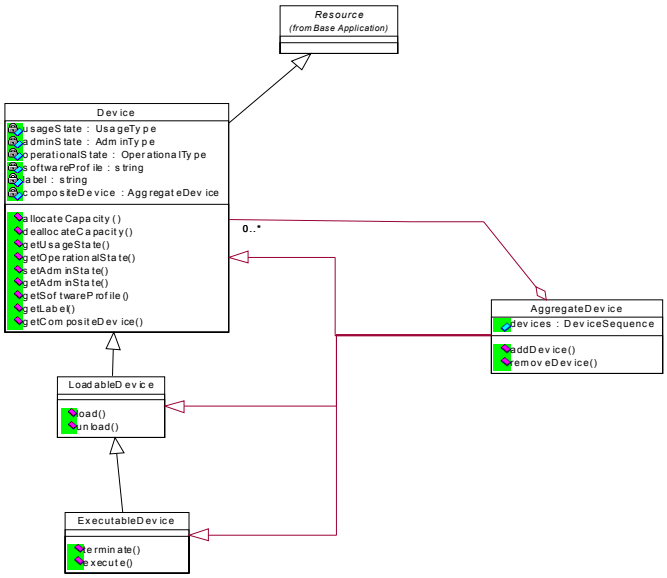
A Device is a type of Resource within the domain. The abstract class Resource defines capabilities and attributes for any logical Device in the domain. A logical Device is a functional abstraction for a set (e.g., zero or more) of hardware devices and provides the following attributes and operations:

1. Software Profile Attribute – This SPD XML profile defines the logical *Device* capabilities (data/command uses and provides ports, configure and query properties, capacity properties, status properties, etc.), which could be a subset of the hardware device's capabilities.

2. State Management & Status Attributes – This information describes the administrative, usage, and operational states of the device.

3. Capacity Operations **-** In order to use a device, certain capacities (e.g., memory, performance, etc.) must be obtained from the *Device*. The capacity properties will vary among devices and are described in the Software Profile. A device may have multiple allocatable capacities, each having its own unique capacity model.

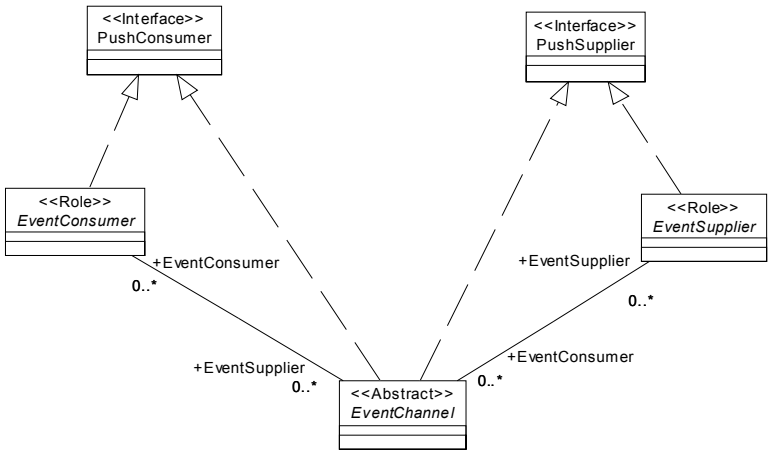
Device Types



CF Services Package

- Event Services
- Logging Services
- File Services
- Naming Services

Event Services



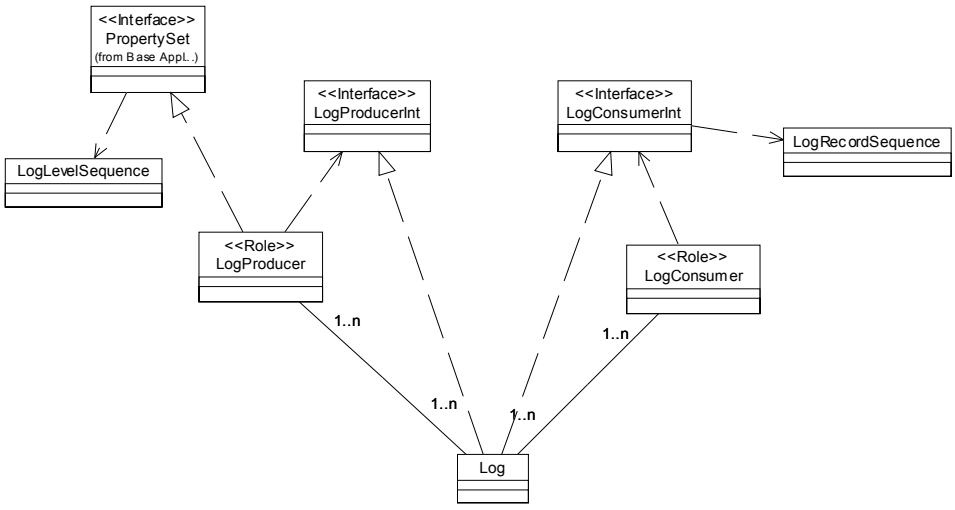
Event Services

The EventChannel represents an abstract concept that can be implemented using different mechanisms. One example is the OMG Event Service implementation, which connects "Proxy" suppliers to consumers and vice versa. However, it is not necessary to implement the EventChannels in the same way, as long as the following requirements are adhered to:

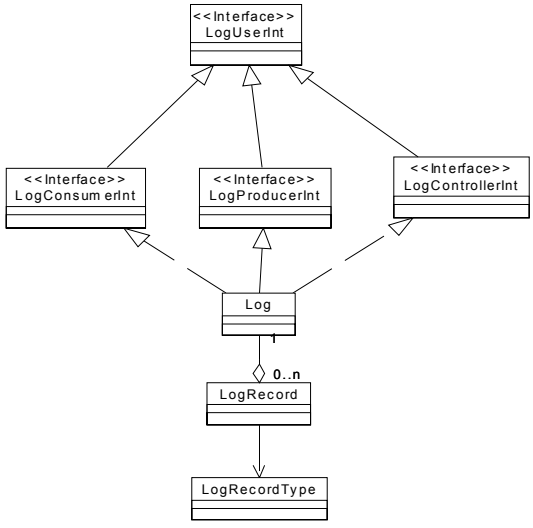
Requirements:

- EventChannels exist if and only if they are named in the Domain Profile (except for the IncomingDomainEventChannel and the OutgoingDomainEventChannel, which always exist)
- Must implement Asynchronous communication
- Decouples EventSupplier and EventConsumer
- Implements the PushConsumer Interface to communicate with EventSuppliers
- Implements the PushSupplier Interface to communicate with EventConsumers

Logging Services



Logging Services: Log Relationships



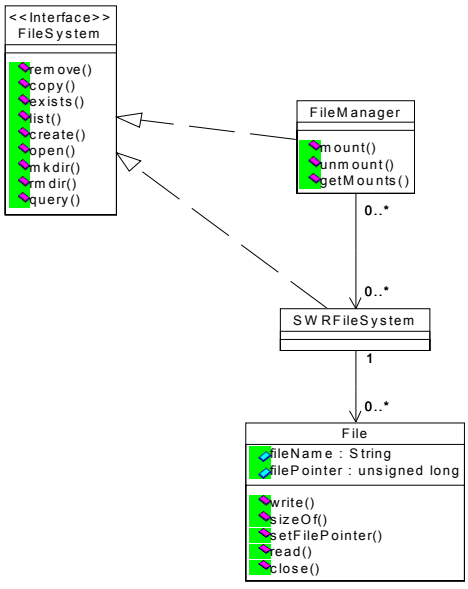
Logging Services

A Lightweight Logging Service is being proposed as an OMG RFC. The OMG Software Radio Special Interest Group is the focal point of this activity. However, the particular characteristics of the proposed Lightweight Logging Service open up a much wider field of application, particularly in the area of embedded and/or real-time systems.

Key characteristics, which distinguish this service also from the OMG Telecom Log Service, are:

- No dependency on other services. The Lightweight Logging Service is a stand-alone service.
- Aim for a minimum footprint. The Lightweight Logging Service contains just a basic time-based logging facility. No expensive components like filters, federation, or event forwarding are contained to keep the size of an implementation to a minimum.
- The logging system specified by the Lightweight Logging Service is totally decoupled from potential log producer clients. Emitting log records to the logging facility has no side-effects on the operation or the timing of the log producer.

File Services



The file service classes (FileManager, FileSystem, and File) are used for installation and removal of application files within the system, and for loading and unloading application files on the various processors that the Devices execute upon.

Naming Services

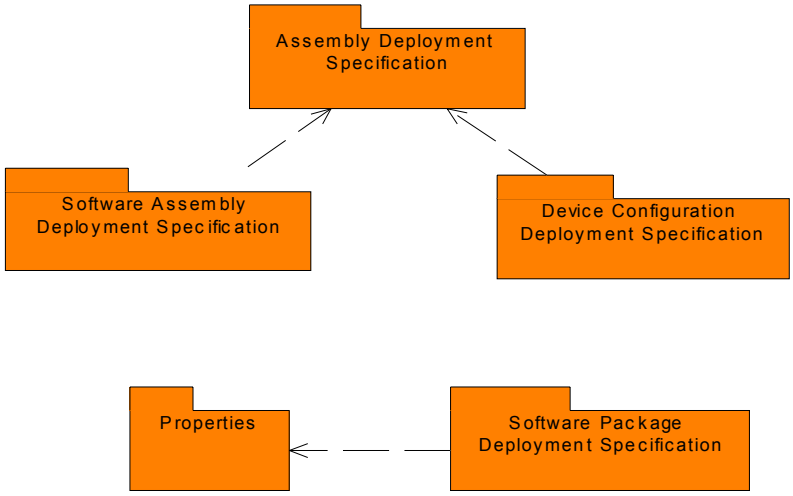
A Naming Service shall be provided in the OE (e.g. a CORBA Naming Service). The Naming Service supplied by an OE shall support the following operations: bind, bind_new_context, unbind, destroy, and resolve. These operations shall meet the requirements of OMG Document formal/00-11/01: Interoperable Naming Service Specification. An example of such naming service is the CosNaming CORBA module and its NamingContext interface's operations.

A Naming Service's NameComponent structure is made up of an id-and-kind pair. The "id" element of each NameComponent is a string value that uniquely identifies a NameComponent. The "kind" element of each NameComponent shall be "" (null string).

Domain Profile Package

The hardware devices and software components that make up an SCA system domain are described by a set of files that are collectively referred to as a Domain Profile. These files describe the identity, capabilities, properties, inter-dependencies, and location of the hardware devices and software components that make up the system. All of the descriptive data about a system is expressed in the XML vocabulary. For purposes of this SCA specification, the elements of the XML vocabulary have been based upon the OMG's CORBA Components specification (orbos/99-07-01). [Note: At the time of this writing, 99-07-01 is a draft standard].

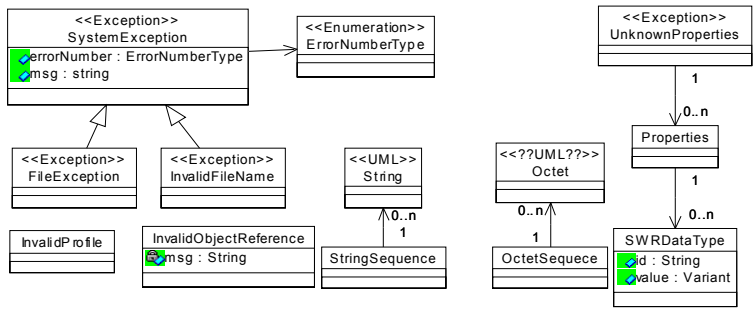
Domain Profile Package



CF Base Types Package

- A utility package that defines data types used by other elements of the UML model of the SWR

CF Base Types Package



1. Introduction to Software Radio
2. Introduction to UML
3. Software Radio UML Model
4. Work in Progress

Current Projects/Collaborations

- Simulator for the SCA specification (Mercury Computer Systems, CRC , CITO)
- SDR Metamodel (Mercury Computer Systems, Mitre, Raytheon)
- Platform Independent Model (PIM) (Mercury Computer Systems and CRC)
- SCA Validation Framework (Mercury Computer Systems)