PRISMTECH
Productivity Tools & Middleware

OMG SBC Workshop: Realizing the Vision

OMG Software Radio Specification Tutorial

Presented by: *Jerry Bickle*

Date: *March 5th 2007*

Agenda

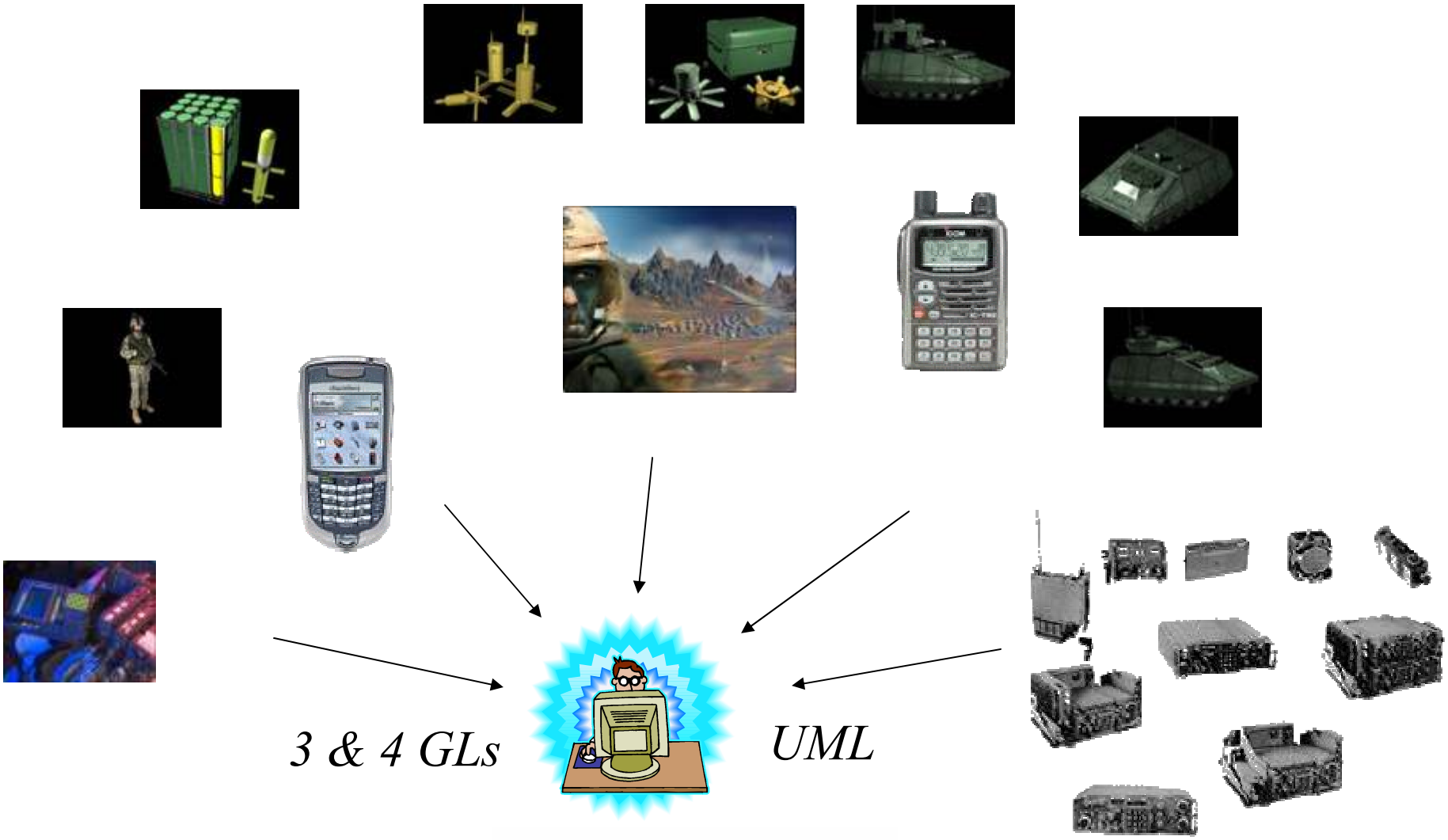
- Software Radio Specification Intention
- Software Radio Specification
- SCA Relationship
- Conformance

Software Radio Specification Intention

- To define a common language for building Software Radios for commercial and military that can be extended and/or constrained, and transformed and implemented in any technology.
- To enable the use of Model Driven Architecture (MDA) and Model Driven Development (MDD) technologies for developing radio systems to address today's problems:
  - General Purpose programming languages and tools have not kept pace with the growth of platform complexity[1]
  - Families of systems are the order of the day
  - Systems and requirements have become more complex
  - Exchange information with other tools in a standard format.

[1] *Model Driven Engineering, Douglas C. Schmidt, IEEE Computer Magazine, February 2006*

Insufficient tools to do the Job for SDR Platform Complexities



SDR Platform Complexities

- Architectures
- Radio Functionality
- Building SDR

SDR Platform Complexities – Architectures[2]

- Adaptive Radio
- Hardware Radio
- Cognitive Radio
  - Intelligent Radio
  - Policy-Based Radio
- Reconfigurable Radio
  - Software Controlled Radio (SCR)
  - Software Defined Radio (SDR)
    - Software Radio (SR)

[2] *IEEE P1900™/D01 - Standard Definitions and Concepts for Spectrum Management and Advanced Radio System Technologies, 16 November 2006 (v 0.25)*

SDR Platform Complexities – Radio Functional Capabilities[2]

- Adaptive Modulation
- Radio Awareness
  - Location Awareness
- Cognitive Control Mechanism
  - Requires Radio Awareness
- Frequency Agility
- Policy-Based Control
- Software Controlled
- Software Defined
- Transmit Power Control

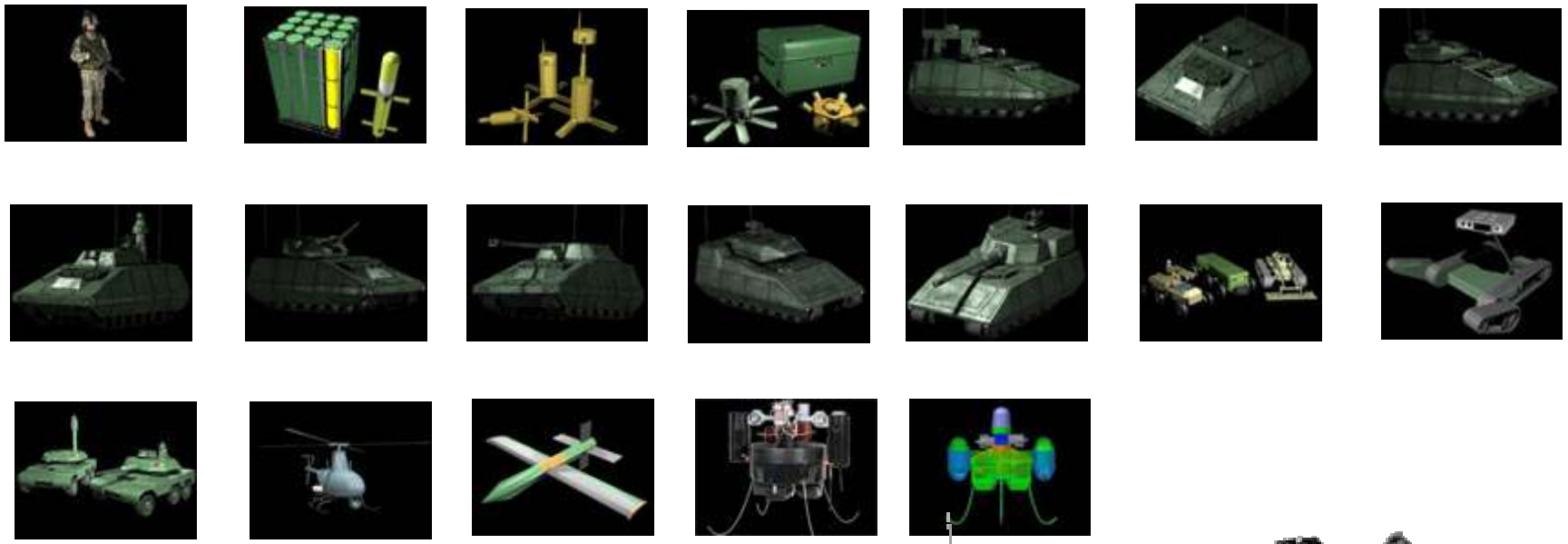
[2] *IEEE P1900™/D01 - Standard Definitions and Concepts for Spectrum Management and Advanced Radio System Technologies, 16 November 2006 (v 0.25)*

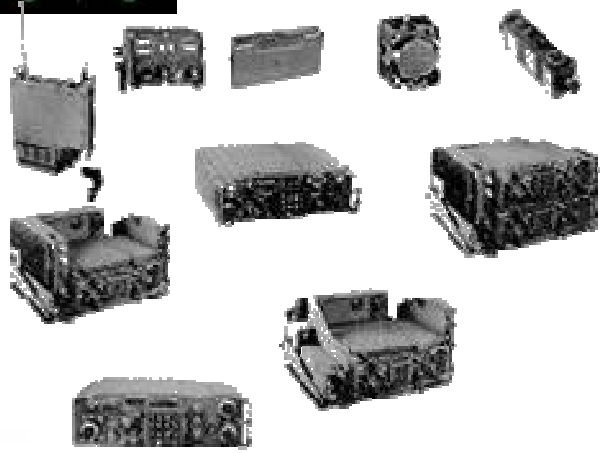
SDR Platform Complexities – Building SDR

- Object Oriented
- Component Based
- Multithreaded/MultiProcess
- Real-time
- Embedded
- C++/Java/Ada/VHDL
- Platform Independent
- High Performance
- Heterogeneous
- Distributed
- Vital
- Secure
- Fault Tolerant
- Portable
- Standardized
- Declarative
- Imperative

- Design Patterns
- Coding Guidelines
- CORBA
- XML
- Compliancy
- Testing
- Multiple Tool Sets

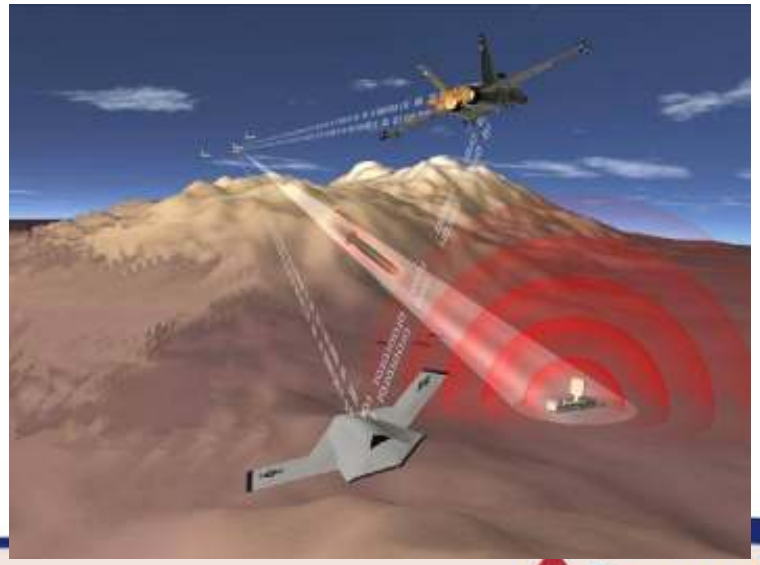
Families of Systems





http://www.army.mil/fcs/

More Complex Systems and Requirements



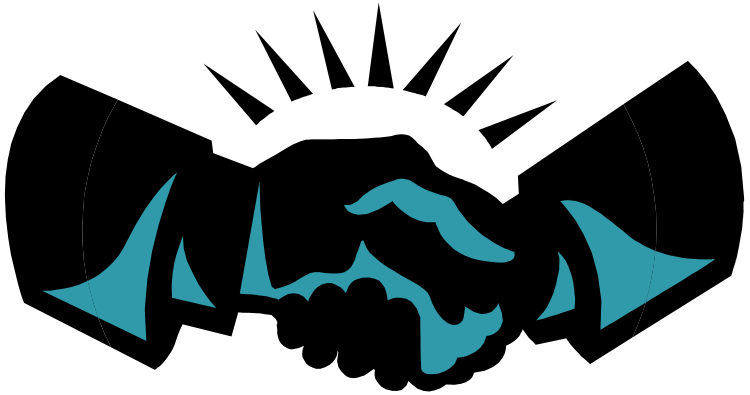
Model Driven Development

- Brings Modeling and Programming together
- Brings System and Software Engineering together.
- Brings Domain Specific Modeling Language tool together with existing tools

MDD – Modeling and Programming Together

Bringing modeling and programming together

Modeling



Programming

"Instead of forcing a separation of the high-level engineering/modeling work from the low-level implementation programming. It brings them together as two well-integrated parts of the same job"[3]

[3] *Eclipse Modeling Framework, Budinsky et.al Addison Wesley 2004*

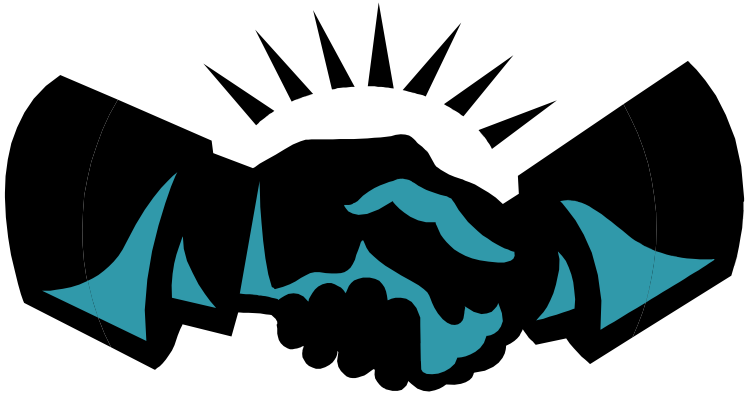
Some Ramifications

- Modeling and programming blend into one activity
- Models become *development* artifacts - not just *design* artifacts
- Domain Specific Models can be machine processed and thus integrate well with generators
- Developers can “program in terms of their design intent rather than the underlying, computing environment”[1]

MDD- Systems and Software Together

Bringing Systems and Software together

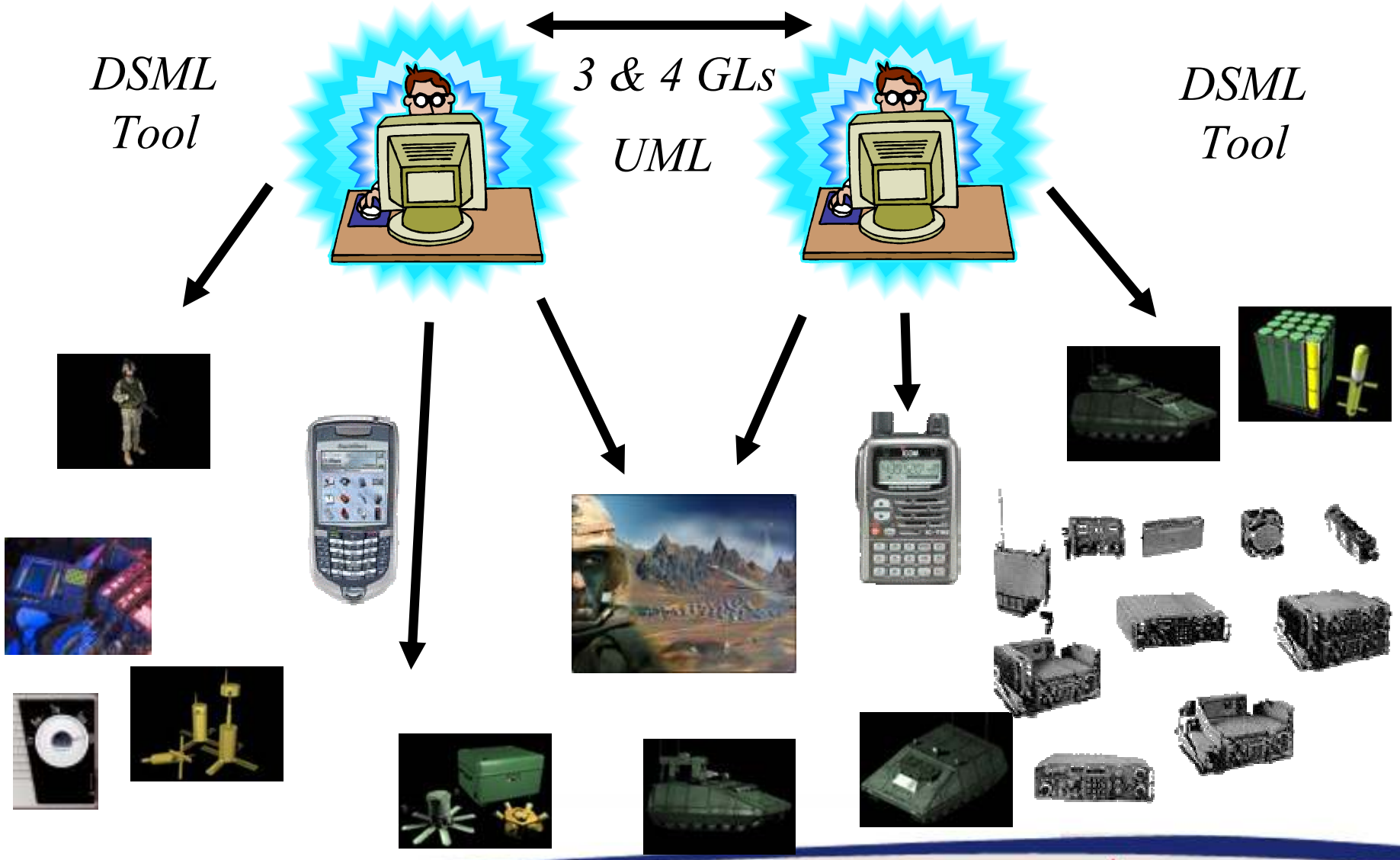
Systems Engineering



Software Engineering

Increased cross discipline collaboration
Speaking the same language
Flattens learning curves

Providing sufficient tools to do the job



PRISMTECH
Productivity Tools & Middleware

Model Driven Development/Generative Programming

- Model Driven Development (MDD)/Generative Programming (GP)
  - The process of moving from a higher level abstraction to a lower level abstraction *automatically*
  - Definition of transformation rules support this paradigm

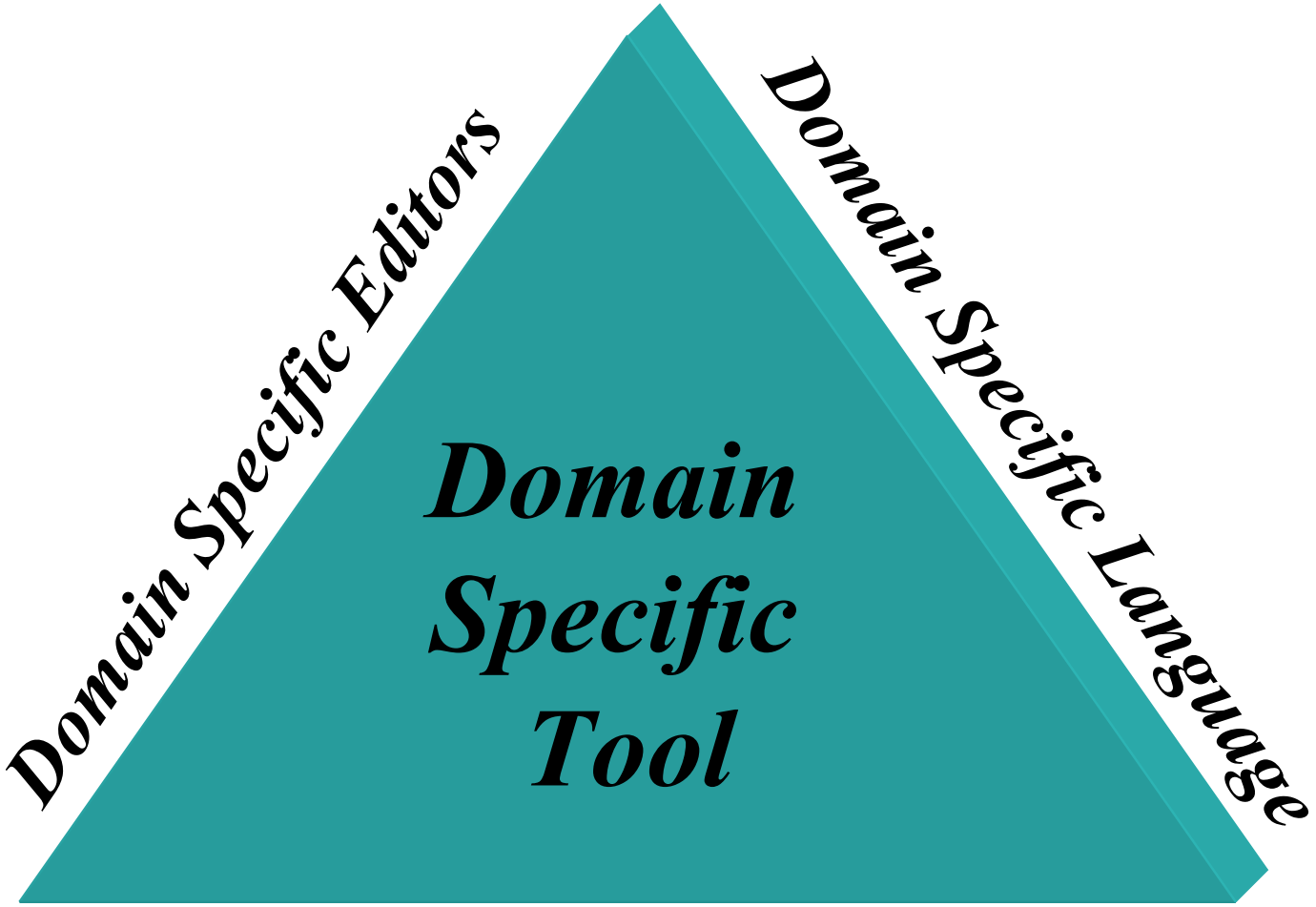
    e.g. C++ to Assembly to Opcodes.
  - Domain Specific Models and Languages work in concert with generative technologies thereby increasing productivity dramatically

MDD Features

- Domain Specific Modeling Languages[1]
- Domain Specific Editors
- Transformation engines and generators

- Combine the above three – Called Model Driven Development (a.k.a. Model Driven Engineering, or Model Integrated Computing)

MDD Features

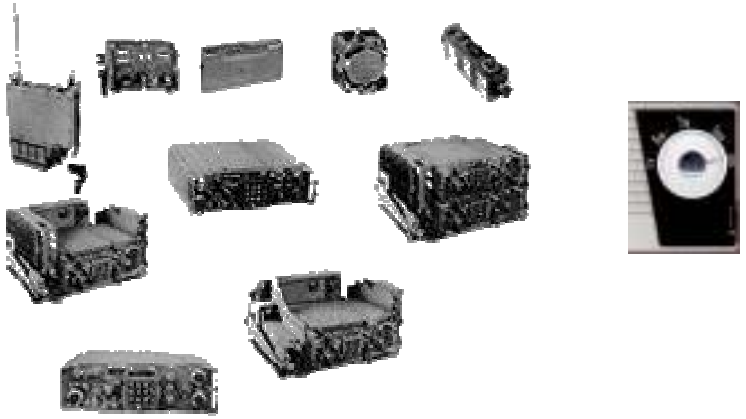


Domain Specific Language

- Focus on System Families allows one to identify the commonalities and variablities found across family members
- And develop DSLs to:
  - ***Factor out*** common behavior into paramaterizable abstractions
  - Provide ***extension mechanisms*** to incorporate variation points found across family members
- And further develop generators to synthesize concrete functionality for a particular family member

Software Radio DSL - Radio Families

Radio Family Members



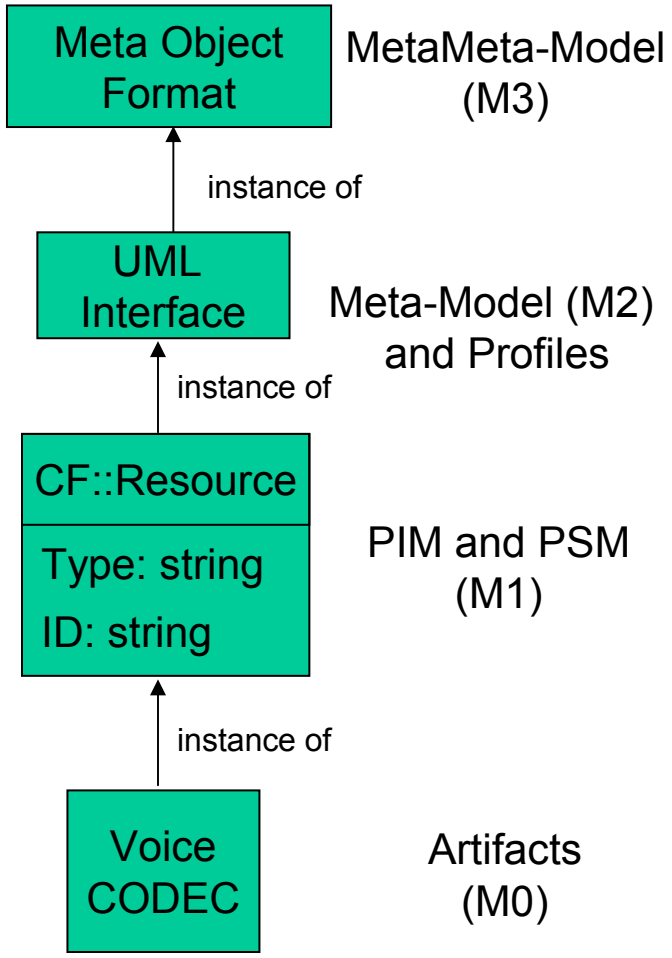
- ***Commonalities***
  - Properties
  - Tests
  - Life Cycle
  - Communications Path
  - Deployment
  - Functionality (Routing, Networking)
  - Basic architecture
- ***Variabilities***
  - Functionality
  - RF or SiS characteristics
  - Processing Elements (HW)
  - Size weight and power constraints

What is a Domain Specific Language?

- In order to understand DSLs, one must understand levels of modeling
- DSLs are defined using Meta-Models
- Meta-Models are defined using even higher level models

So, a Domain Specific Language is…

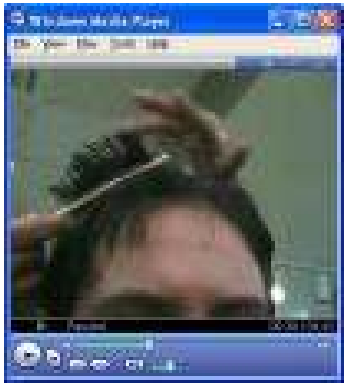
- A language targeted to a particular problem
  - Such as Software Radios
- Not a general purpose language aimed at any kind of problem
  - Such as UML
- DSL Example, OMG PIM and PSM for Software Radio Components
  - UML Profile for Software Radio



Domain Specificity

Domain Independent

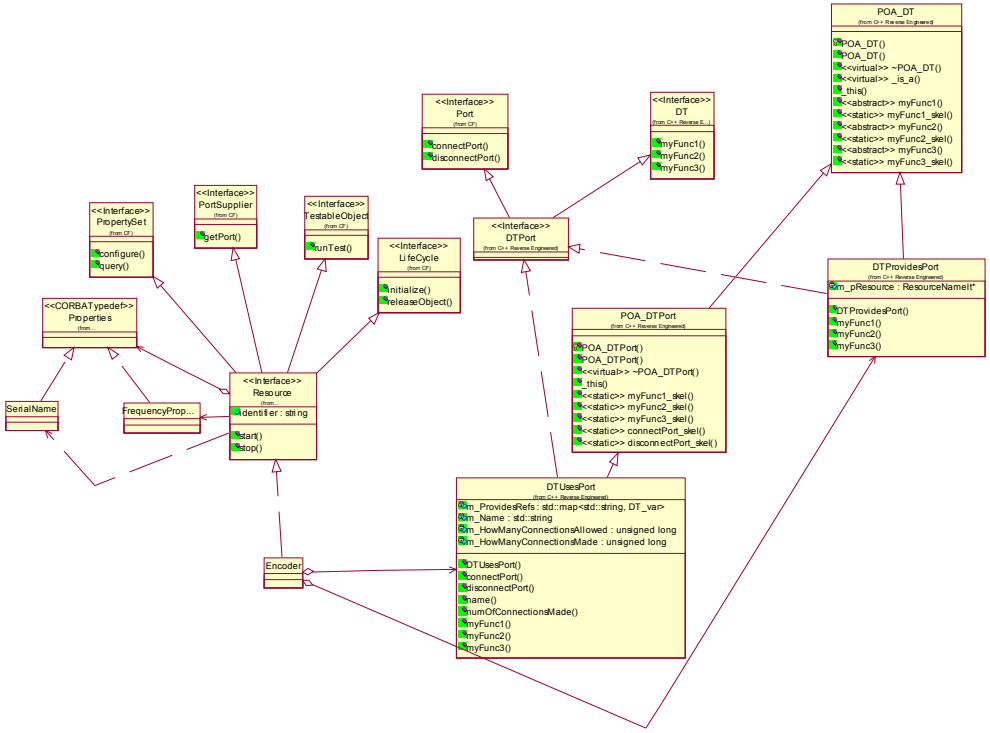
Domain Specific - Tools



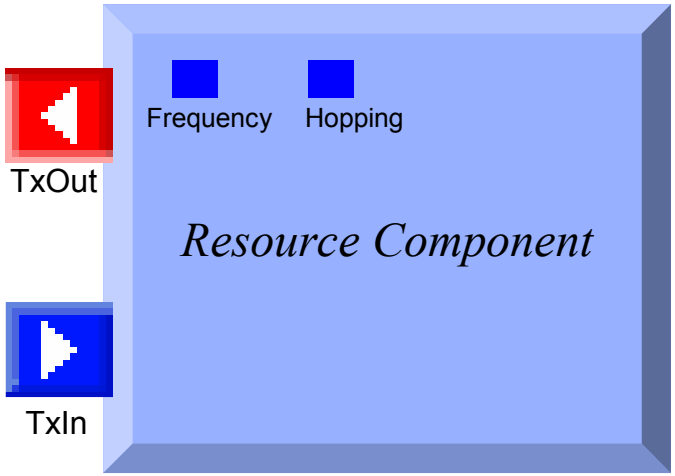
The task at hand

Domain Specific Modeling

- DSLs allow simplified modeling in the ***Platform Independent Model Design*** vs. complex modeling ***in the Solution Space***



Solution Space Modeling



PIM Design Modeling

Single Resource Component with 2 Ports and 2 Properties

PRISMTECH
Productivity Tools & Middleware

Declarative Specification/Programming

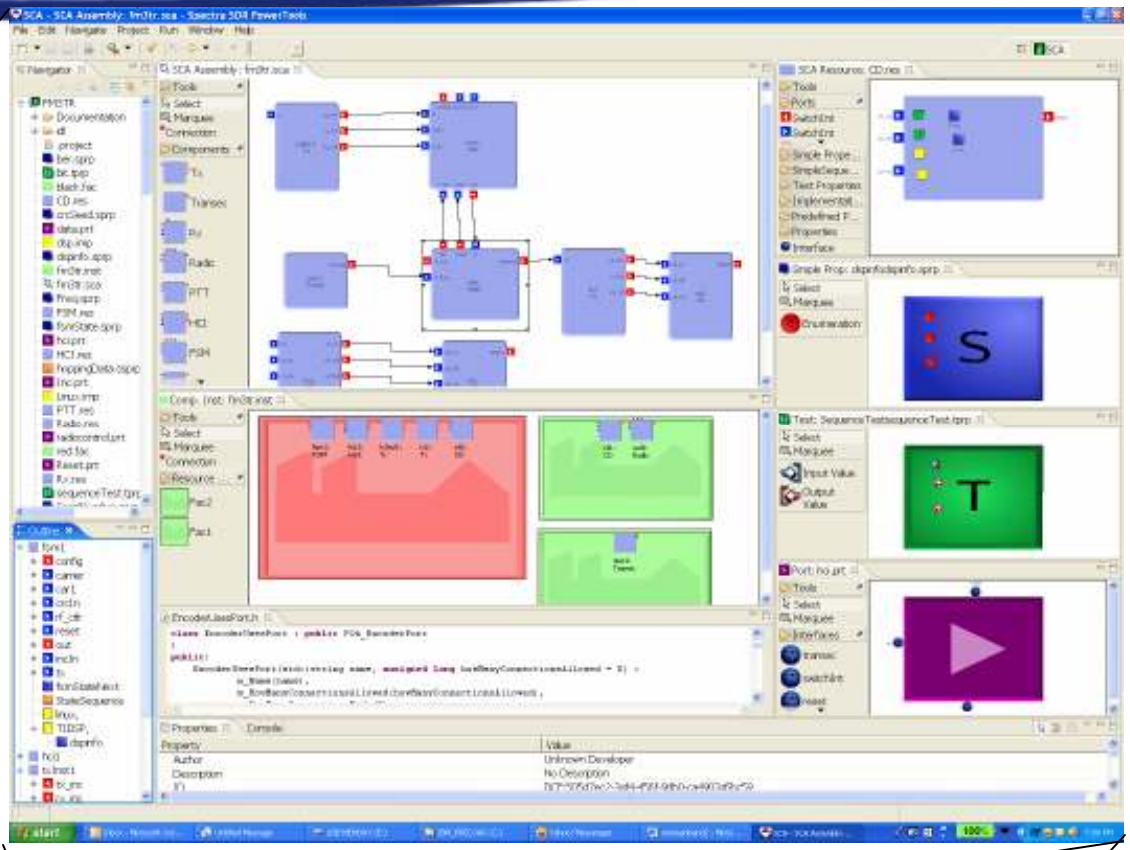
- Both the DSL and GDSL are *declarative* in nature
- Involves programming by setting properties, making connections and establishing relationships
  - Versus imperative sequential procedural instructions[1]
- "Declarative programming improves productivity and quality because it is another form of reuse of preprogrammed, prevalidated logic"[3]

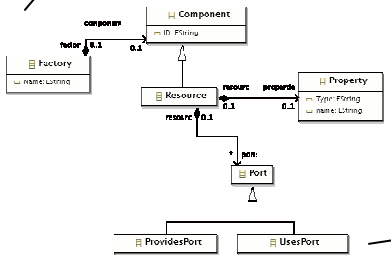
[3] *Model Driven Architecture D. Frankel, Wiley 2003*

A Graphical Domain-Specific Language (GDSL)

Images, layout, organization based on meta-model

GDSL





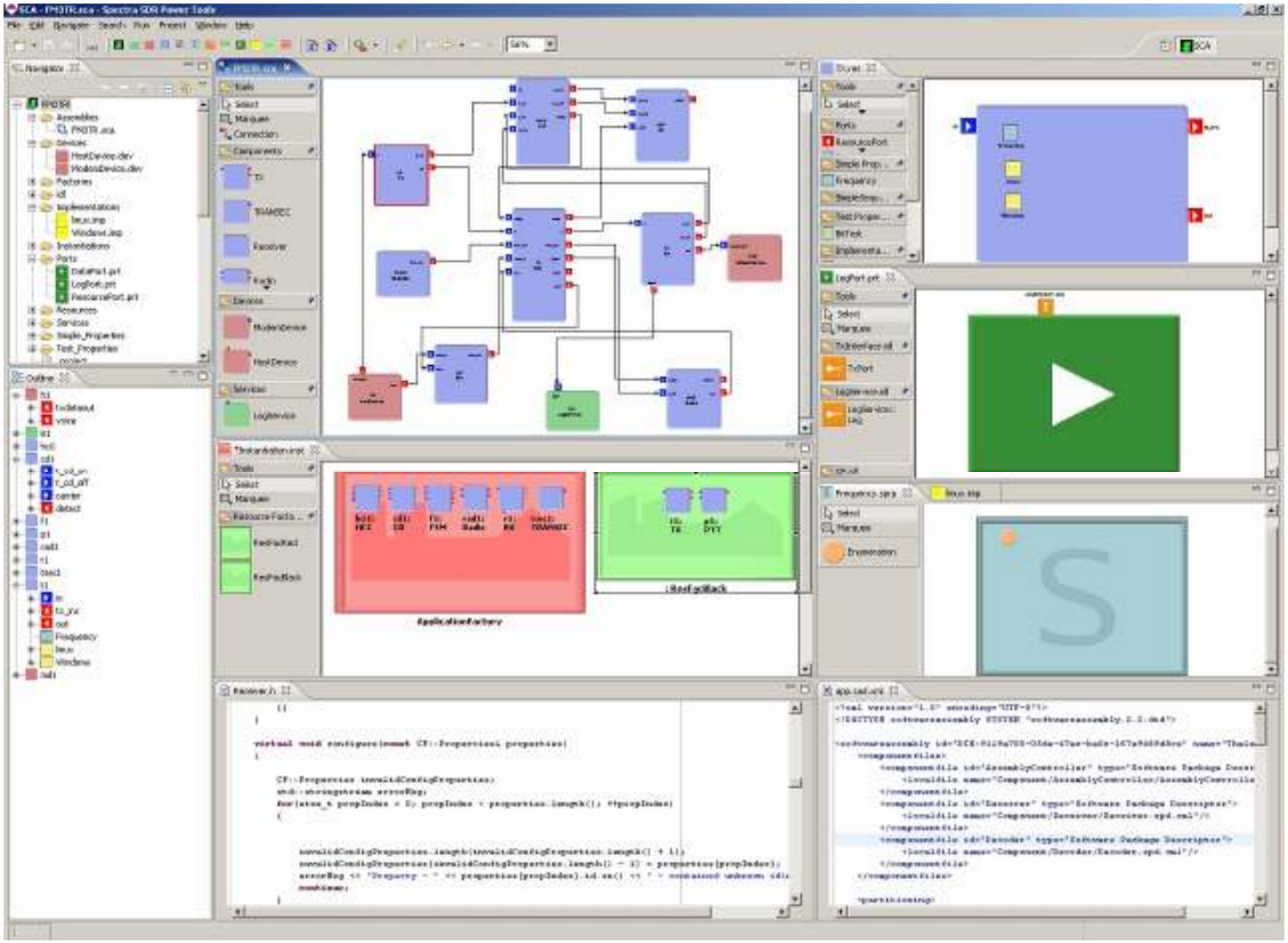
DSL

```
<components Name="BitFlipper" organization="PrismTech"
id="DCE:8f647411-91a1-4295-bbc6-6d3eff4982f7">
<ports xsi:type="com.prismtech.spectra.sdr.sca2_2.models:UsesPort"
    instanceName="TX" name="Data"/>
<ports xsi:type="com.prismtech.spectra.sdr.sca2_2.models:ProvidesPort"
instanceName="RX" name="Data"/>
</components>
</com.prismtech.spectra.sdr.sca2_2.models:Assembly>
```

PIM

PRISMTECH
Productivity Tools & Middleware

GDSL Illustrations

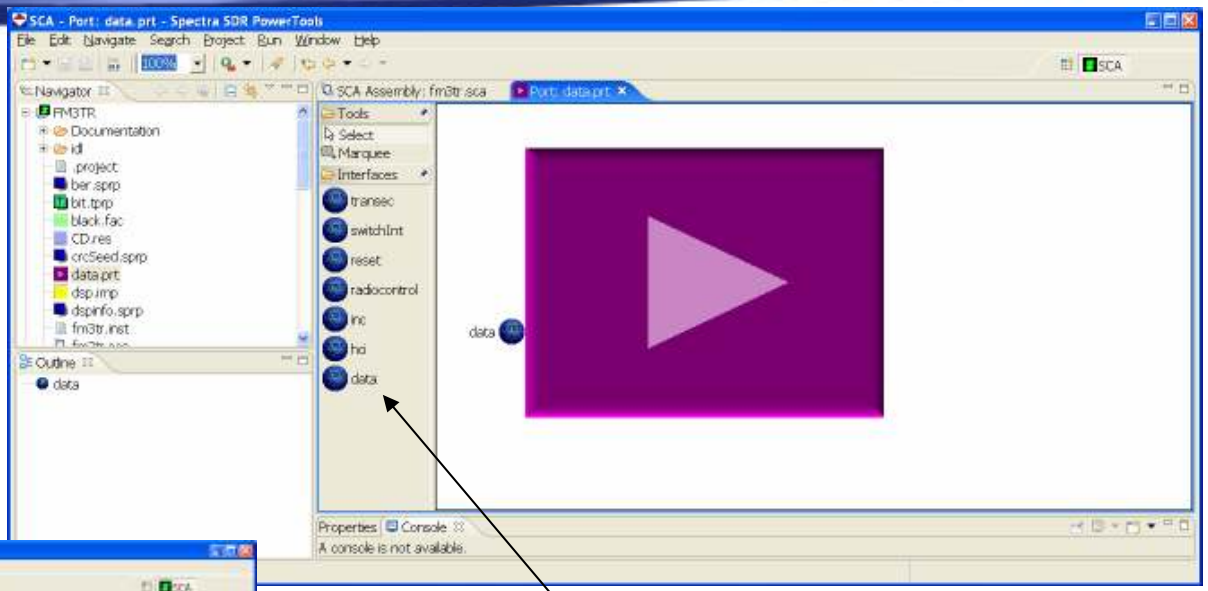


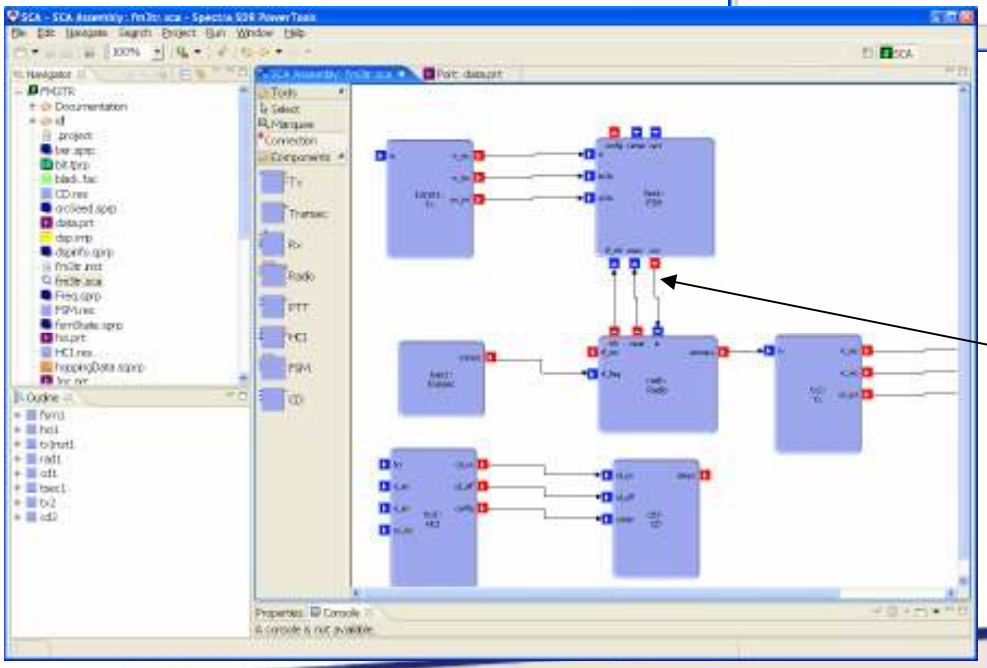
PRISMTECH
Productivity Tools & Middleware

Processing the Declarative Specification

- Constraining its input
- Interpreting it as it is entered
- After interpretation, have the declarative specification drive code generators which will transform the model into an executable form for a given platform

Domain Specific Editors - Constraints Enforcement





Enforce structural compositional, directional, etc constraints. Preconditions, postconditions and invariants

PRISMTECH
Productivity Tools & Middleware

Generative Programming

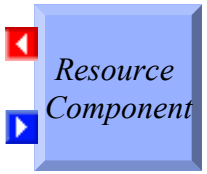
- The process of moving from a higher level abstraction to a lower level abstraction *automatically*
- Specification of transformation rules support this paradigm
  - e.g. C++ to Assembly to Opcodes.
- Domain Specific Models and Languages work in concert with generative technologies
  - The whole is the worth more than the sum of the parts
- Thereby increasing the productivity

Software Enabling Technologies Levels of Abstraction

Programming | *Platform*

Less Flexible

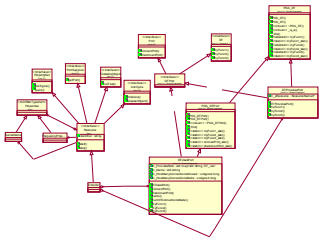
MDD/GP
Component
Development



Frameworks

More Productive

Flexibility



Middleware

Libraries

OOP &
Higher Order Languages
Assembly
Opcodes
Custom Hardware

Operating
Systems

Productivity

More Flexible

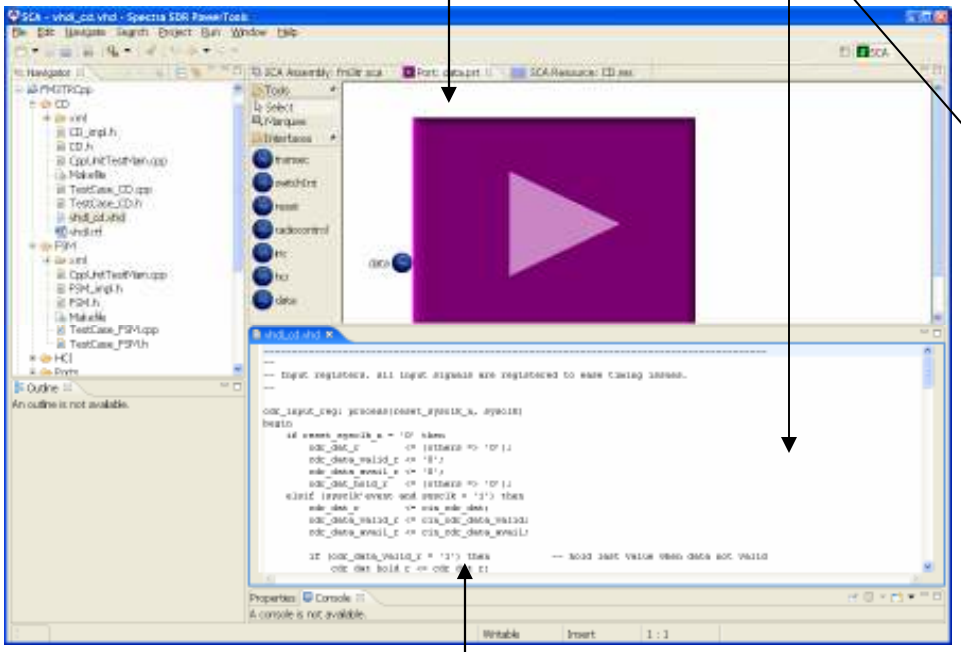
Less Productive

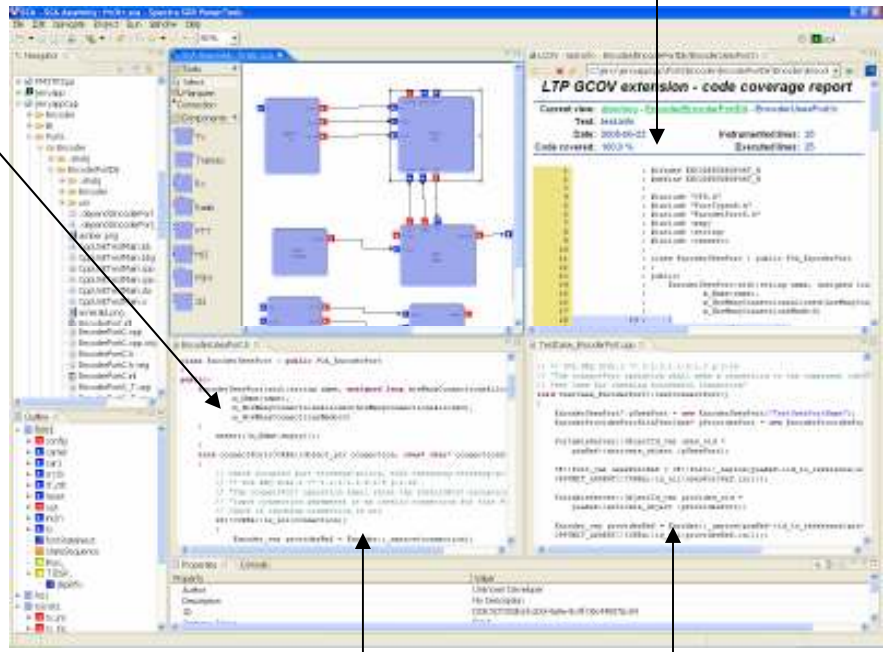
PRISMTECH
Productivity Tools & Middleware

Domain Specific Generators

Translate from declarative to imperative

Code Coverage





VHDL

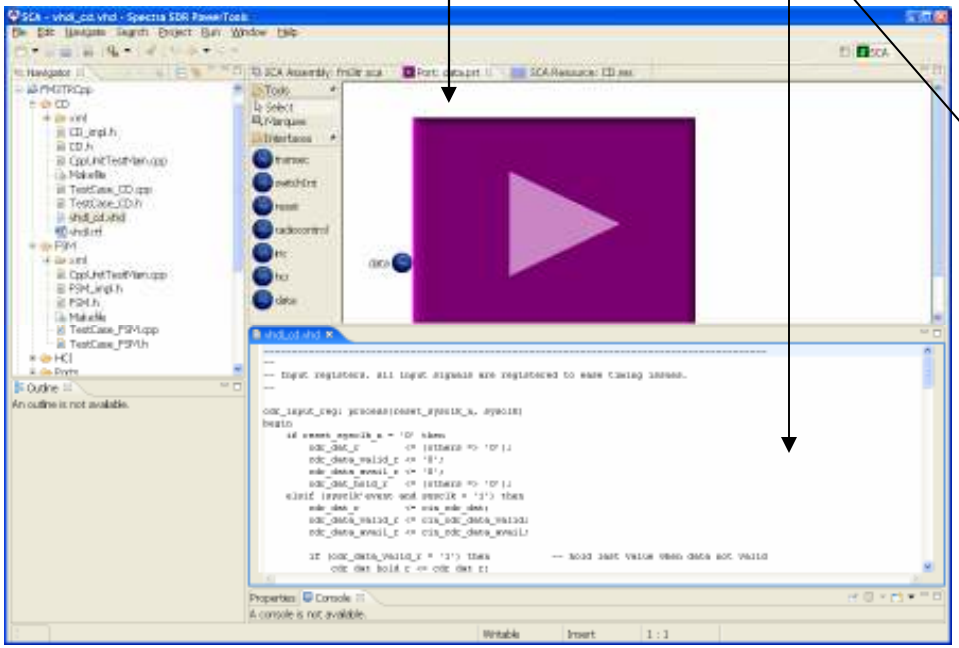
C++

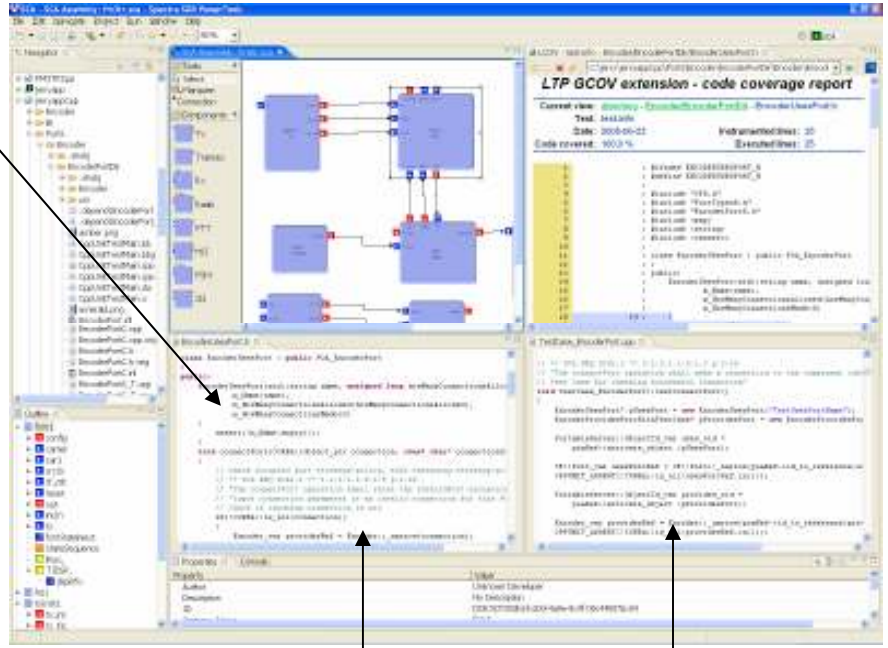
Test Cases

PRISMTECH
Productivity Tools & Middleware

Domain Specific Generators, continued

Application of Design Patterns Automatically





- **Replicated production of Design Pattern(s) implementations**

C++

Test Cases

- **Can be more effective than dealing with the replication using 3GL language features, particularly in DRE Systems**

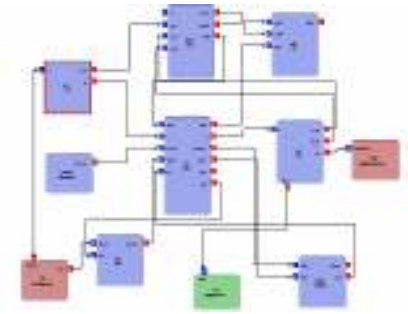
PRISMTECH
Productivity Tools & Middleware

What Does MDD Do For You?

Model Driven Development

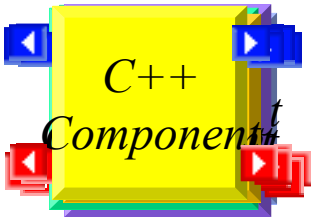
Tools should be designed specifically to support the SW Radio domain

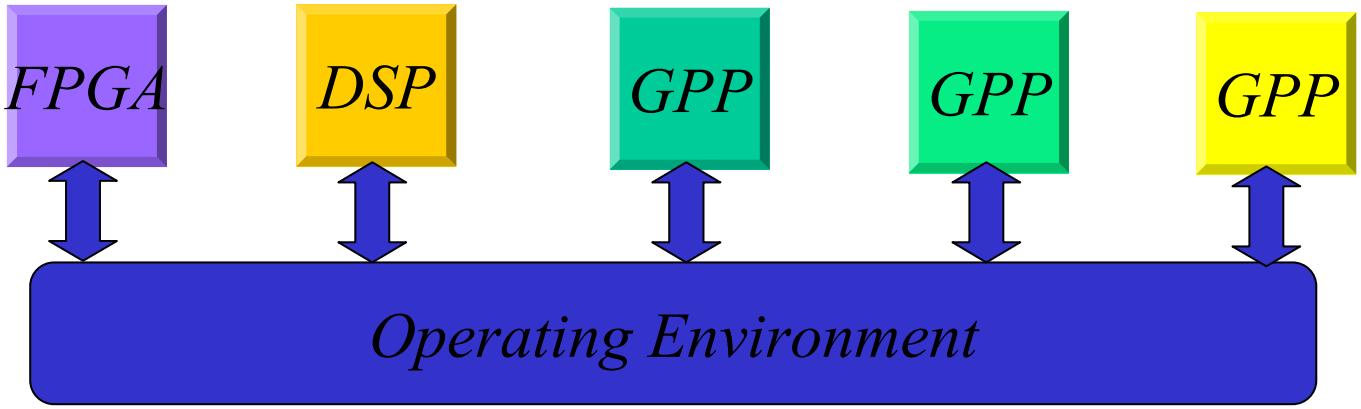
Not just simple computer aided S/W engineering tools



Transform the Model directly into what you need

MDD Component PIM to PSM





Evolution and Transformation

- A SDR MDD Tool should have the capability to support different technology transformations
- A SDR MDD Tool should have the capability of model to model transformation

MDD Summary

- Consist of Platform Independent Model (PIM) and Platform Specific Model (PSM) views
- PIM (using a GDSL Tool) is used to capture the design of an application independently on any technologies.
  - So the Application/Business Logic can be applied against any technology
- The PSM captures technologies and implementation artifacts transformed from a PIM using a GDSL
  - Descriptor Files (XML DTD, Schema)
  - Source Files (C, C++, Java, VHDL)
  - Executable Files
  - Detailed Design to specific language

PRISMTECH
Productivity Tools & Middleware

Software Radio Specification

Spectra Software Defined Radio Products

Software Radio Specification

- Software Radio Specification
  - Software Radio Domain Specific Language
    - Component Framework Profile
      - *Component Framework Specification Volume*
    - Communication Channel Profile
      - *Communication Channel and Equipment Specification Volume*
  - Software Radio Facilities
    - Component Framework Facilities PIM
      - *Component Framework Specification Volume*
    - Common and Data Link Layer Facilities PIM
      - *Common and Data Link Layer Facilities Specification Volume*
    - Physical Layer Facilities PIM
      - *Communication Channel and Equipment Specification Volume*
  - Platform Specific Technologies
    - Component Document Type Definitions Specification Volume
    - POSIX Profiles Specification Volume

Note: the Information being presented represents the Revision Task Force (RTF) changes.

Software Radio Normative XMI Files

- Software Radio Profile.xml
  - Component Framework Profile.xml
  - Communication Channel Profile.xml
  - Common and Data Link Layer Facilities.xml

Dependencies

- Communicaiton Channel Profile has dependencies to Component Framework, and Common and Data Link Layer Facilities.
- Common and Data Link Layer Facilities has dependencies to Component Framework

Software Radio Domain Specific Language

- Software Radio DSL in conjunction with a GDSL tool, provides the capability of building and designing Waveform applications and Platform services as PIMs independently of a specific technology and transforming the PIMs into platform specific technologies.

Software Radio Domain Specific Language

- UML Profile mechanism is used to specific the Software Radio Domain Specific Language (DSL) since UML already has existing elements such as:
  - Artifact
  - Component
  - Device
  - Interface
  - Port
  - Property

Software Radio DSL - UML Profile

- UML Stereotypes were used to extend the UML Elements with specific semantics and constraints.
  - Object Constraint Language is used to express constraints where practical
- The profiles also contains M1 level elements in model library packages
  - These are mostly interfaces that the component definitions are constrained to.
- The profile is captured in XMI, which allows for GDSL tool to exchange information in a standard format.
  - Work that is being completed in the RTF

Software Radio DSL - UML Profile

- The profile consists of:
  - Terminology
  - Constraints
  - Rule Set
  - Relationships
  - Requirements
- Which can be extended and/or constrained for specific domains
  - Handset
  - Base Station
  - Space

PRISMTECH
Productivity Tools & Middleware

Software Radio Specification

Component Framework (CF) Profile

Spectra Software Defined Radio Products

Component Framework Profile

- CF Application Components
- CF Infrastructure Components

PRISMTECH
Productivity Tools & Middleware

Software Radio Specification

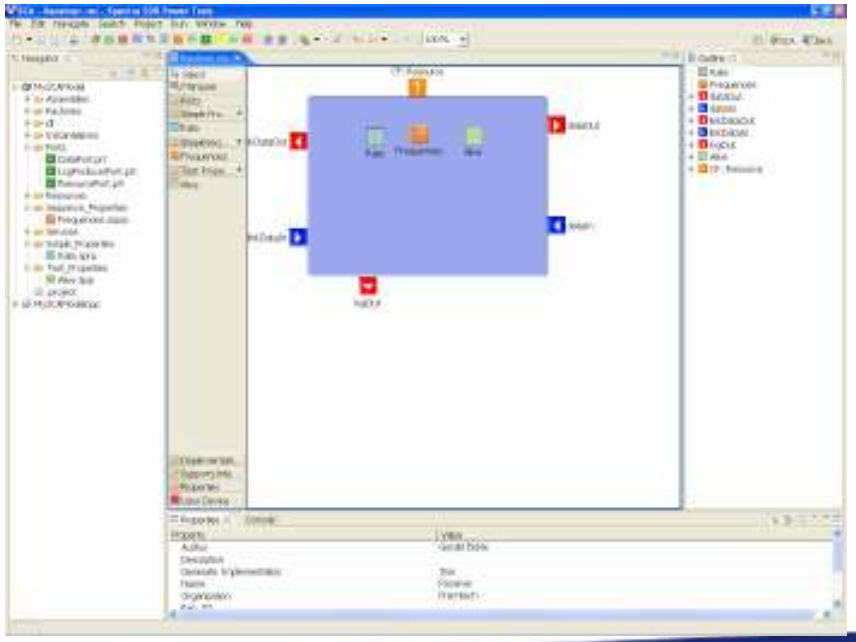
Component Framework (CF) Profile

CF Application Components

Spectra Software Defined Radio Products

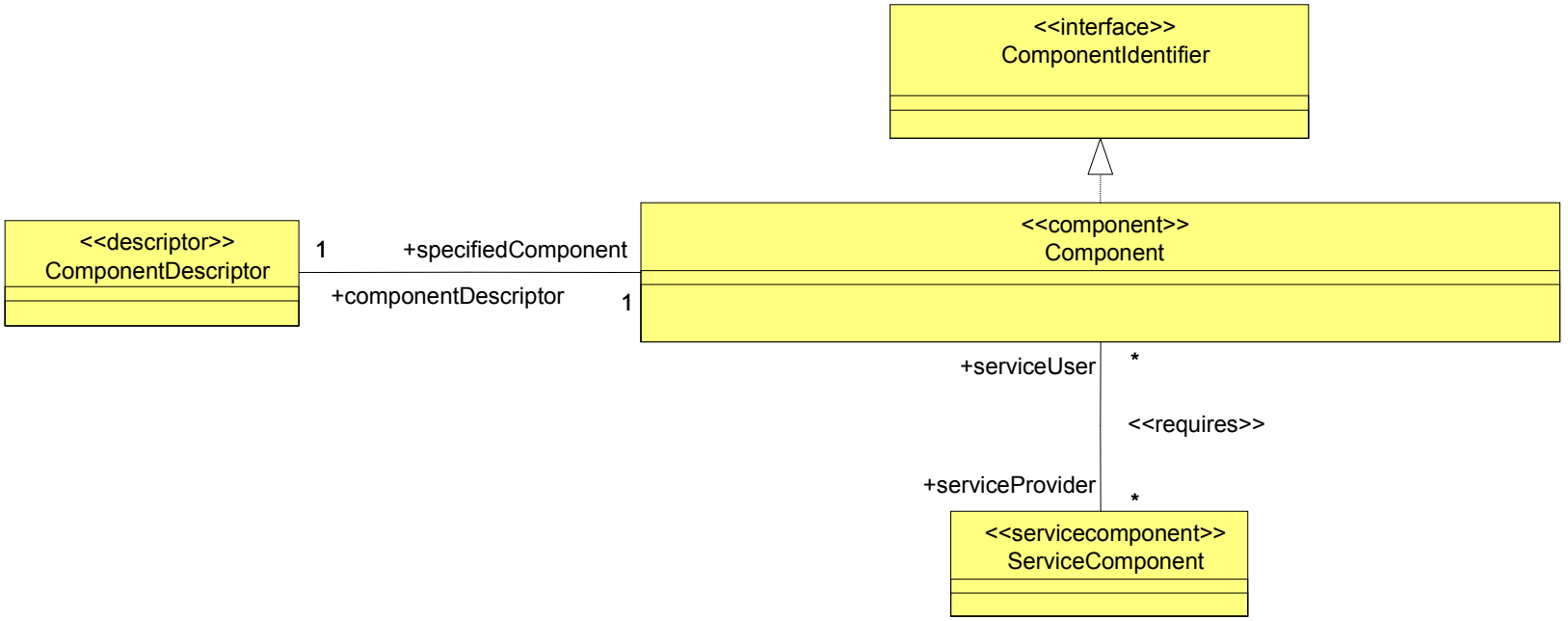
Component Overview

- Component
  - Can realize interfaces, Support Type Interfaces
  - Interface Properties
  - Ports
    - Provides Port - provides an interface implementation
    - Uses (or Requires) Port - uses an interface that is implemented by another component



Software Radio Component Definition

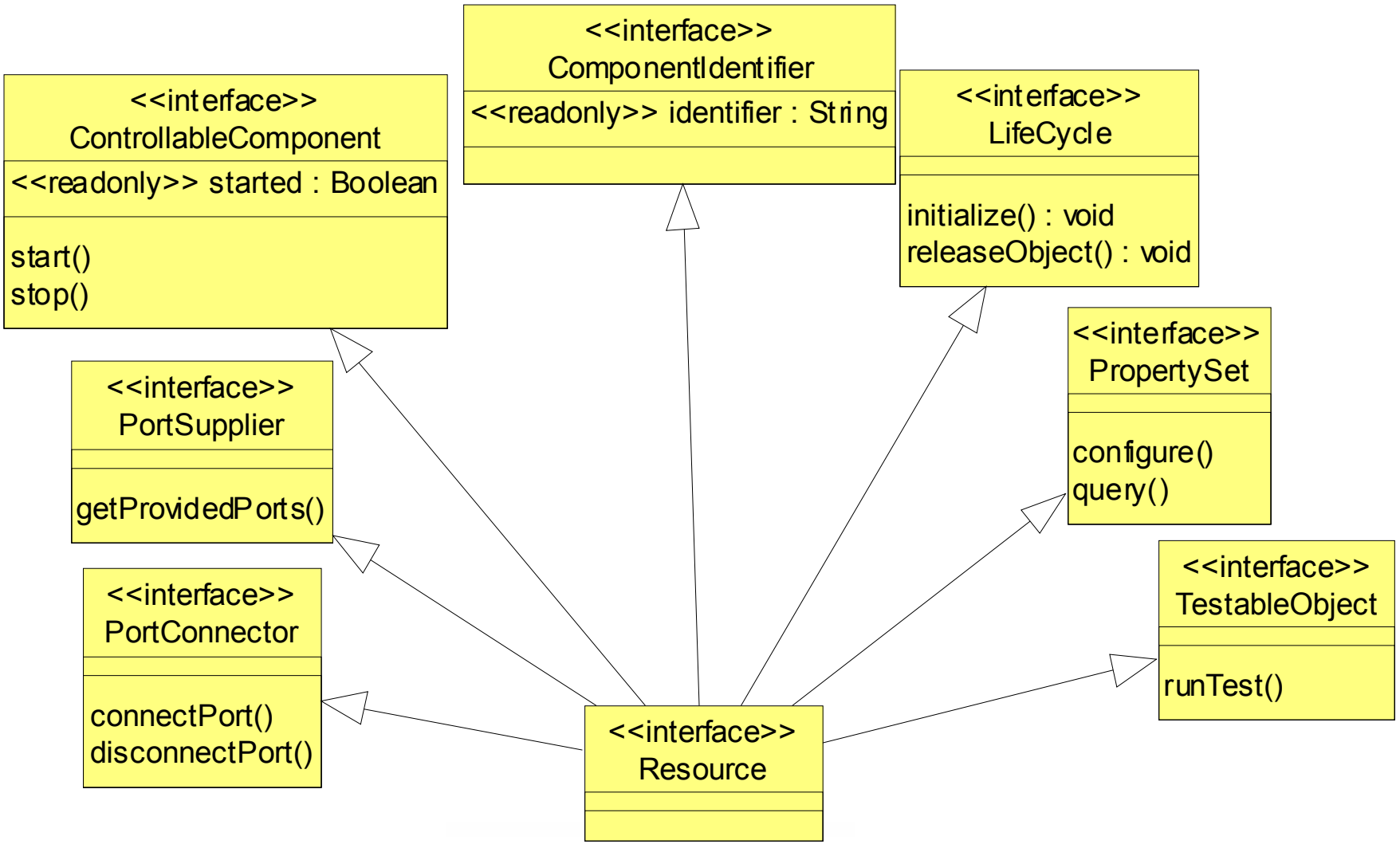
- Software Radio Component
  - Extends the UML Component by having a unique identifier



Software Radio Component Base Interfaces

- A set of ***minimal*** generic interfaces for component management that not only support deployment behavior
  - Lifecycle – life cycle management for a component
  - PropertySet – generic configure and query operations that are based upon primitive types
  - TestableObject – generic interface for testing a component (e.g., BIT)
  - PortSupplier – retrieval of the provided interfaces for a component
  - PortConnector – the management of connection behavior for a component
  - ControllableComponent – overall control of the component
  - ComponentIdentifier – contains the identifier of the component
- The interfaces support
  - Reconfiguration
  - Control
- Resource interface – specialization of all the generic interfaces
  - Provides a complete generic interface for component management and control

Resource Interface

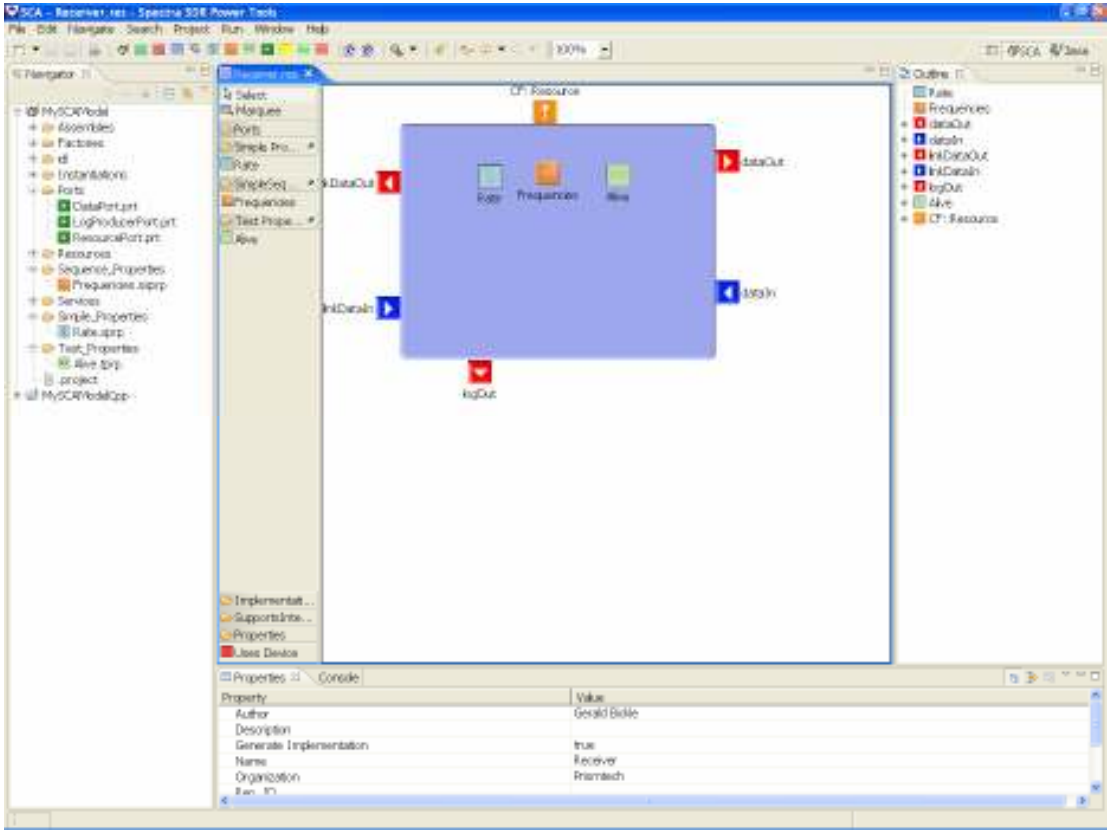


Software Radio Component Interface Properties

- All properties are based upon primitive types
  - Supplement information beyond UML (UML Tags), identifier, range, units of measure, enumeration literals, etc.
- Types of Properties
  - Configure and Query
    - Simple – a property that is a primitive (e.g., octet, boolean, char, string, float, unsigned and signed integers (16, 32, 64 bits), etc.)
    - Simple Sequence – a sequence of simple values of the same primitive type.
    - Struct – a collection of simple properties
    - Struct Sequence – a sequence of struct values of the same struct type.
  - Test
    - Describes a test that contains a set of simple test inputs and simple test outputs
- Each software radio component definition determines its set of properties and interfaces, none mandated by the specification.

Software Radio Component Properties using GDSL Tool Illustration

<<devicecomponent>> SerialIODevice
<<configureproperty>> protocol : UShort
<<configureproperty>> receiveBaudRate : ULong
<<configureproperty>> receiveEncoding : UShort
<<configureproperty>> receiveClockSource : UShort
<<configureproperty>> hardwareFlowControl : Boolean
<<configureproperty>> parityChecking : UShort
<<configureproperty>> characterWidth : UShort
<<configureproperty>> onThreshold : ULong
<<configureproperty>> numberStartBits : UShort
<<configureproperty>> numberStopBits : UShort
<<configureproperty>> receiveBufferSize : ULong
<<configureproperty>> transmitEncoding : UShort
<<configureproperty>> transmitBaudRate : ULong
<<configureproperty>> transmitClockSource : UShort
<<configureproperty>> flowControlXonXoff : Boolean
<<configureproperty>> maxPDUSize : UShort
<<configureproperty>> minPDUSize : ULong
<<configureproperty>> logLevel : LogLevel
<<configureproperty>> ctsStatus : Boolean
<<configureproperty>> rts_cts_mode : Boolean
<<configureproperty>> txActive : Boolean
<<characteristicproperty>> location : String = Red
<<characteristicproperty>> deviceType : String = SerialIO
<<capacityproperty>> inUse : UShort



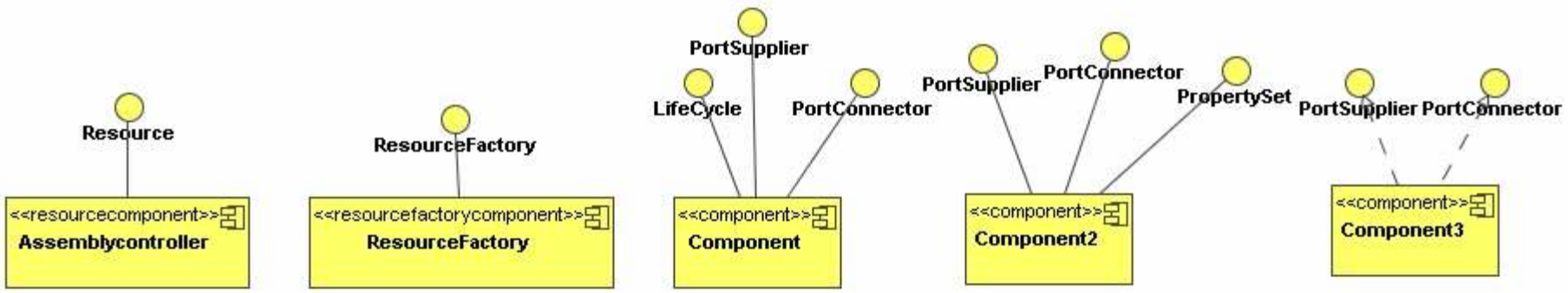
Application Component Definitions

- Component – extension of UML Component that is the base component definition of all software radio components
- Resource Component – a specialization of Component that realizes a Resource interface.
  - The Resource Component is extended by the interfaces (e.g., Common and Data Link Layer Facilities) it provides and/or uses.
- Resource Factory Component – a specialization of Component that is a factory that manages Resource Components

Low Profile Component Definitions

- The OMG UML Profile for SW Radio does not restrict one from building lighter weight SW Radio components other than a Resource Component.
  - Components that have configuration and/or query properties require the property set interface
  - Components that have uses port require the PortConnector interface
  - Components that have provides port require the PortSupplier interface
  - Components that require initialization and teardown require the LifeCycle interface
  - Components that have testable properties require the TestableObject interface
  - Components that require over all control require the ControllableComponent interface

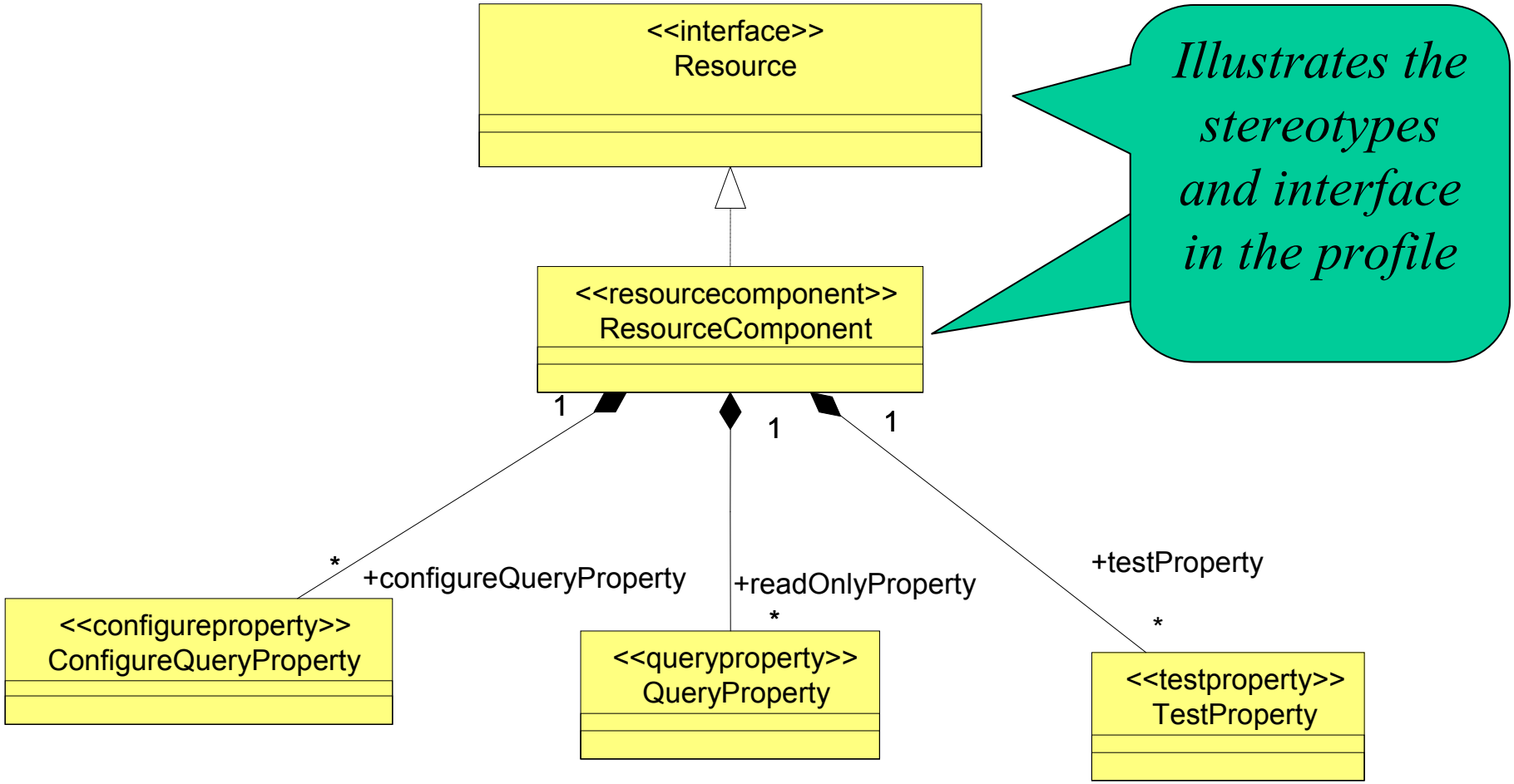
Low Profile Component Illustrations



Resource Component

- A type of Software Radio Component that realizes the Resource interface
  - Supports the Resource interface directly
- Properties
  - Configuration (read and write)
  - Query (read only)
  - Test
- Ports
  - Flow Control
  - Status
  - Data
  - Command and Control

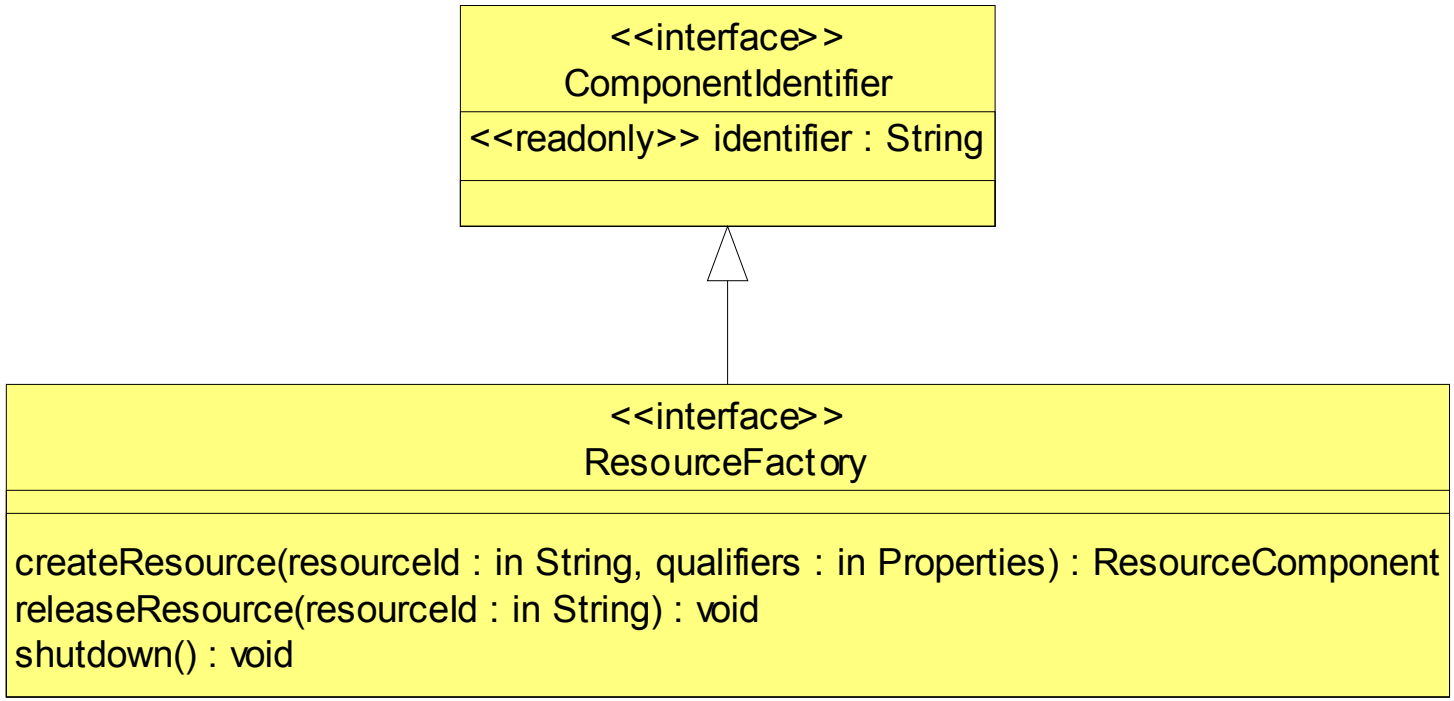
Resource Component Illustration



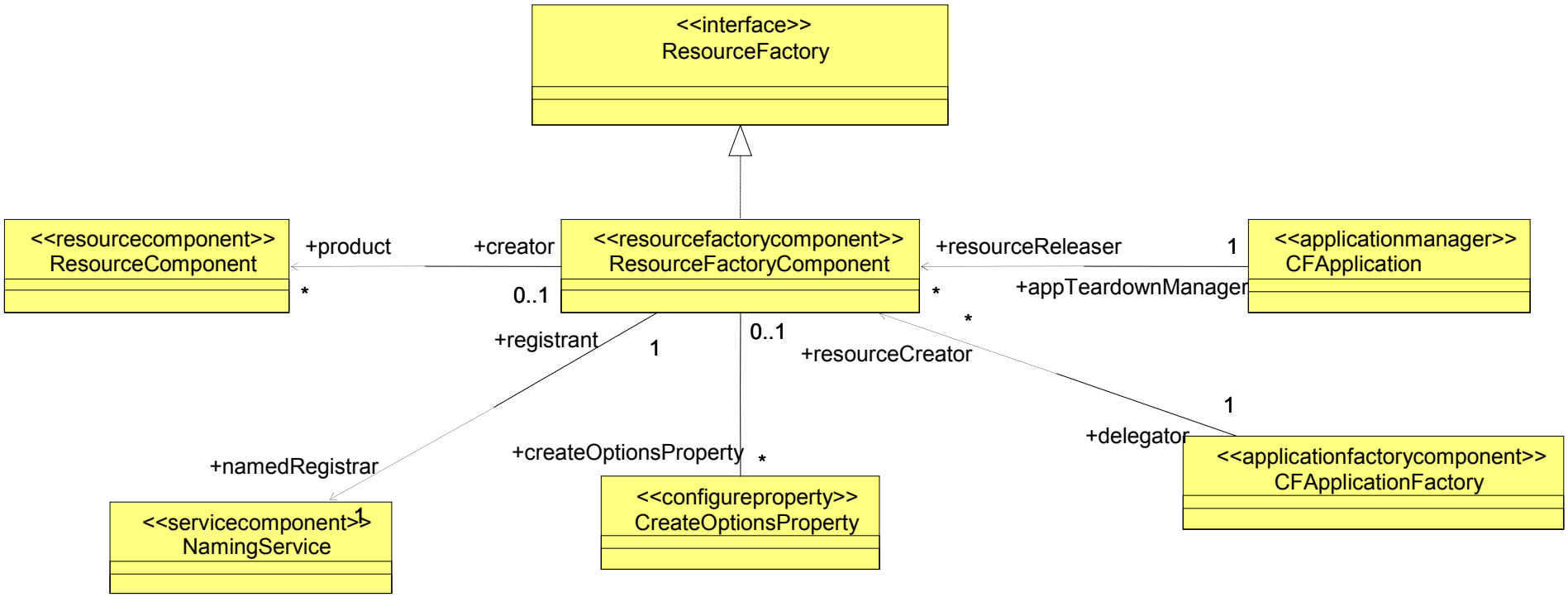
Resource Factory Component

- Realizes the ResourceFactory Interface
  - Used to create/tear down Resource Components
- Behavior can be homogeneous (same Resource Component type) or heterogeneous (different Resource Component types)
- Modeled after the Factory Design Pattern
- Used for collocation of Components within the same process space.
- Optional
  - Waveforms are not required to have these types of components.

Resource Factory Interface



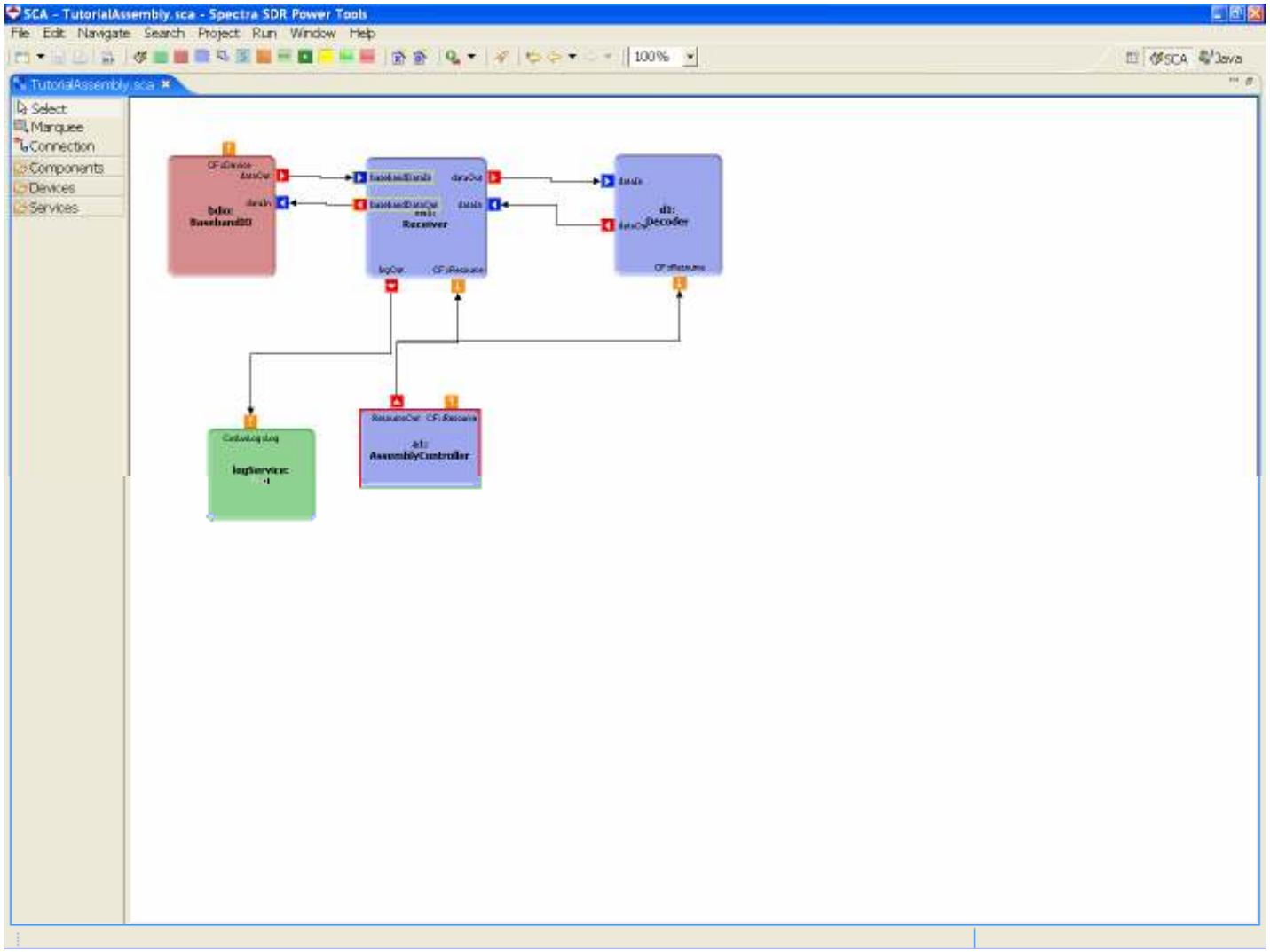
Resource Factory Component Illustration



Application Assembly Component Definition

- Composition of SW Radio Component and/or Resource components and optionally Resource Factory components
  - Resource Factories can be used to create Resource Components in the assembly
- One of the Resource Component(s) is identified as the assembly controller.
  - This is the component a CF Application delegates base interface operations to.
- How SW Radio Components are to be connected together and to what services
- Defines the SW Radio Components ports that are the external ports for the assembly
- Initial configuration values for component instances.

Application Assembly From GDSL Tool Illustration



PRISMTECH
Productivity Tools & Middleware

Software Radio Specification

Component Framework (CF) Profile

CF Infrastructure Components

Spectra Software Defined Radio Products

CF Infrastructure Components

- Domain and Device Management
- Application Deployment Management
- Services

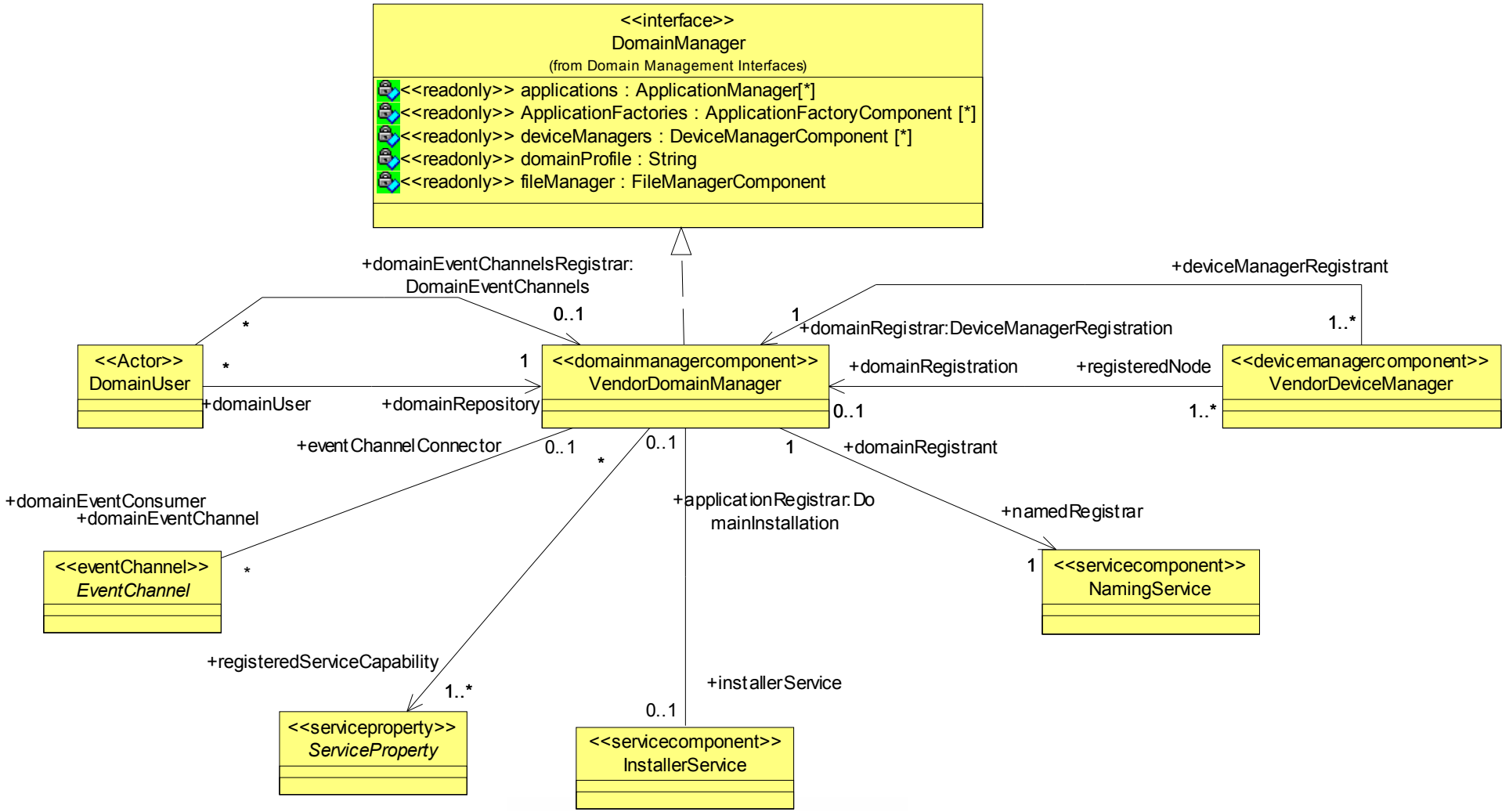
CF Infrastructure Components – Domain and Device Management

- Domain Management – for the control and retrieval of domain information
- Device Management – for the control and retrieval of the common services on communication equipment within a Radio Set

CF Domain Management

- Domain Manager Component realizes the DomainManager interface and provides interfaces that provides:
  - DeviceManagerRegistration interface - provides the mechanism for registering and unregistering DeviceManager Component, and its devices and services within a domain.
  - DomainInstallation interface - provides the mechanism for installing and uninstalling applications within a domain.
  - DomainManager interface - provides the mechanism for retrieval of installed applications, instantiated applications, registered DeviceManager Components, etc.
  - DomainEventChannels interface - provides the mechanism for managing domain's event channels connections.

Domain Manager Component Illustration

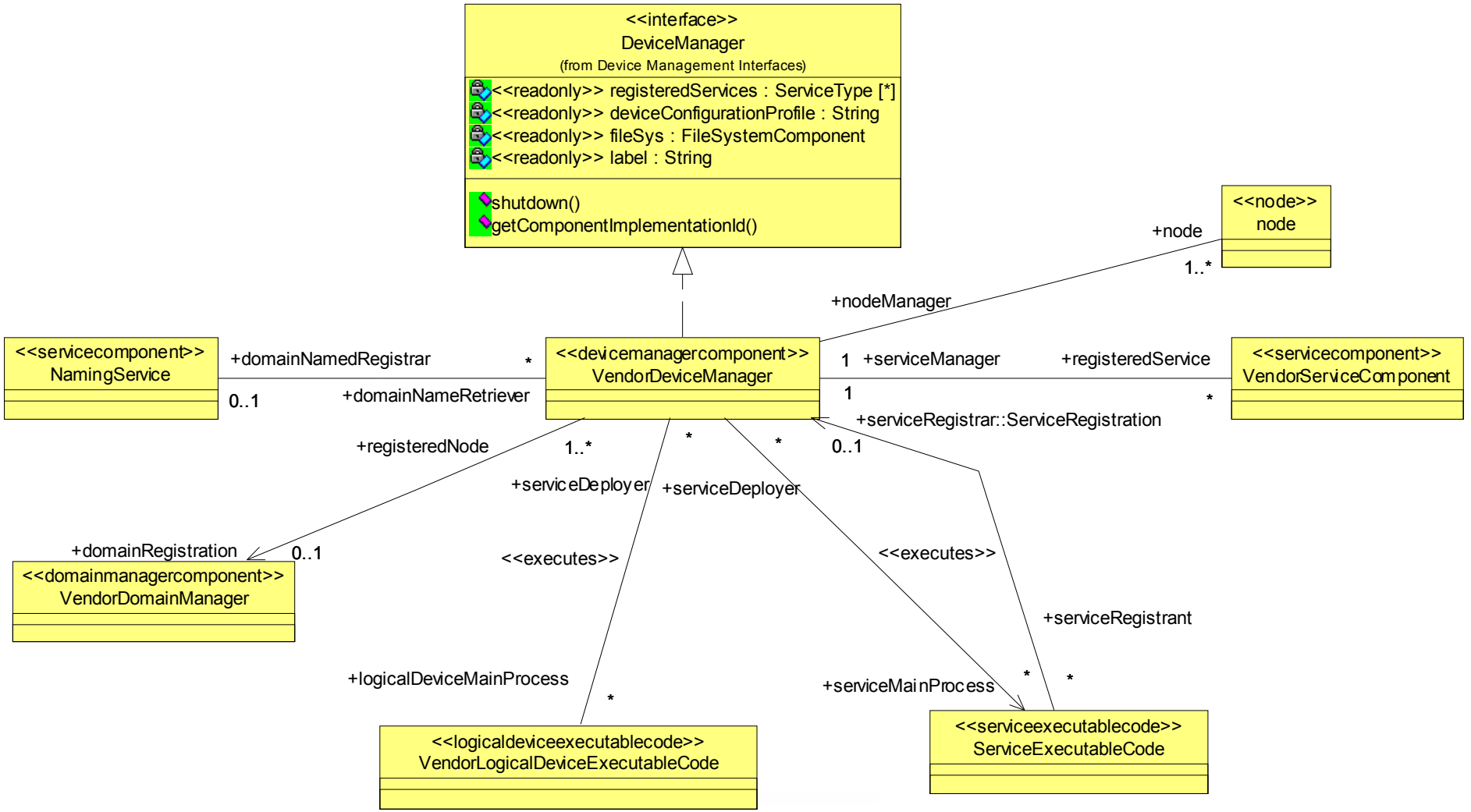


PRISMTECH
Productivity Tools & Middleware

CF Device Management

- Device Manager component realizes the
  - DeviceManager interface
    - UnRegisters/Registers itself and its Devices and services with the Domain Manager component
    - Provides the mechanism for starting up logical Device and Service Components on a device by its descriptor definition
    - Usually has a File System Associated with it
  - Service Registration
    - The capabilities of adding and removing logical Device and Service Components from the environment.

CF Device Management Component Illustration



PRISMTECH
Productivity Tools & Middleware

Software Radio Specification

Component Framework (CF) Profile

CF Infrastructure Deployment Components

Spectra Software Defined Radio Products

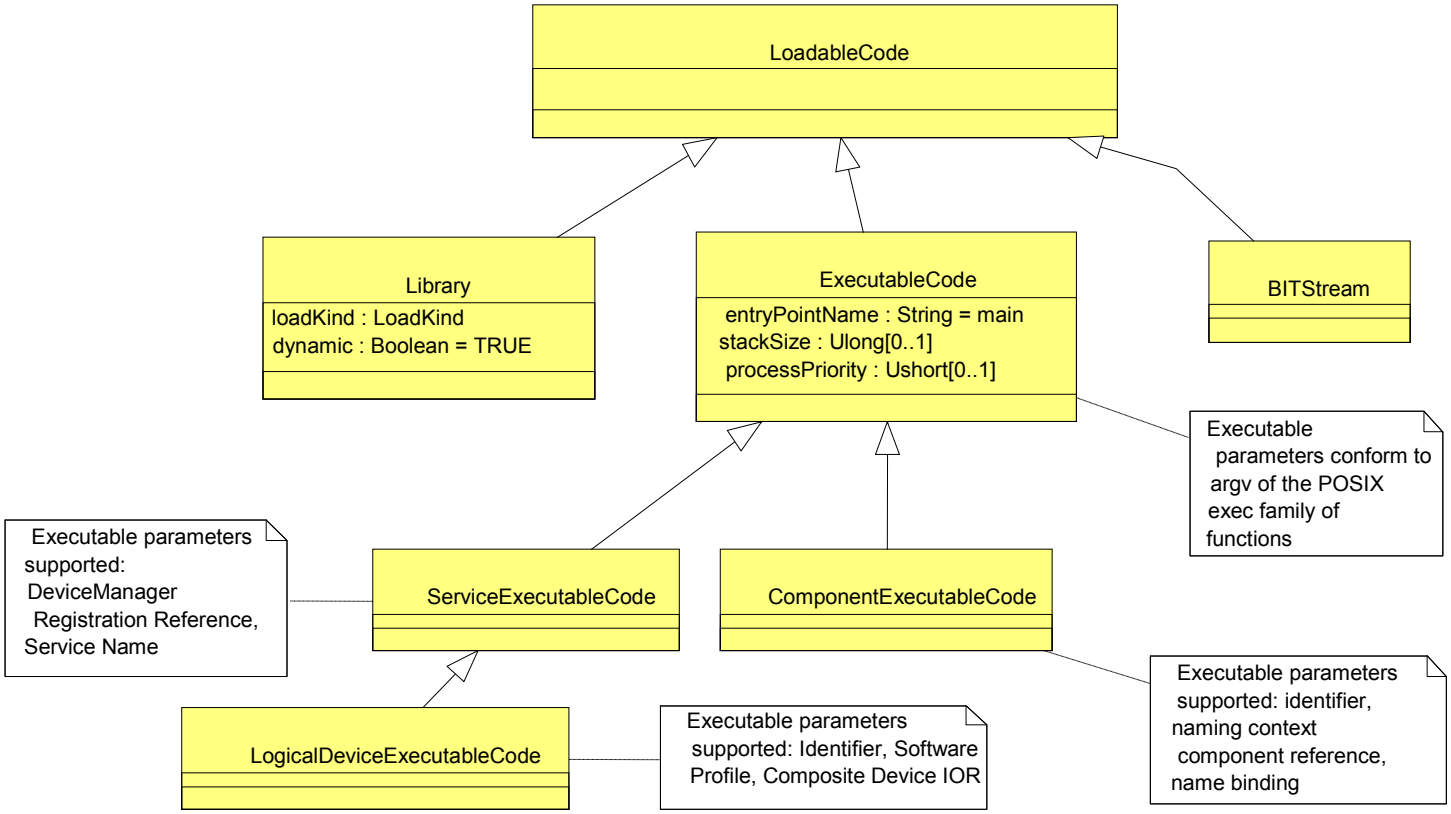
CF Application Deployment Management

- CF Artifacts describes the CF executable artifacts that are involved in the deployment of applications (e.g, waveforms), logical devices, and services within a CF environment.
- CF Applications Deployment Components describes the components that are involved in the deployment of and management of CF Applications

CF Application Deployment – Artifacts

- Loadable Code
  - BITStream is object code for a prologic device (e.g., Field Programmable Gate Array device).
  - Library is loadable static or dynamic object code.
  - ExecutableCode is an executable operating system main process or entry point
    - Logical Device Executable Code is an executable that manifests a Device Component.
    - Resource Executable Code is an executable that manifests either an Resource Component or/andResourceFactoryComponent.
    - Service Executable Code is an executable that manifests a ServiceComponent.
    - User Defined Executable parameters can be used as constructor or environment parameters for the components manifested.
      - *Conform to Argv format*
- Descriptor is a deployment or component specification that conveys information on the element to be deployed

CF Application Deployment – Artifacts Illustration



CF Application Deployment – Descriptor Artifacts

- Descriptors Used For Deployment
  - Software Component Descriptor (SCD) – describes a component properties and port characteristics
  - Software Package Descriptor (SPD) – describes implementations for a component
  - Properties Descriptor – describes properties for a component
  - Software Assembly Descriptor (SAD) – describes an assembly of components that are connected together
  - Device Configuration Descriptor (DCD) – describes a device configuration
- Other Descriptors
  - Domain Manager Configuration Descriptor DMD– describes a DomainManager component configuration
  - Device Package Descriptor (DPD) – describes a hardware device

CF Application Deployment – Applications Deployment Components

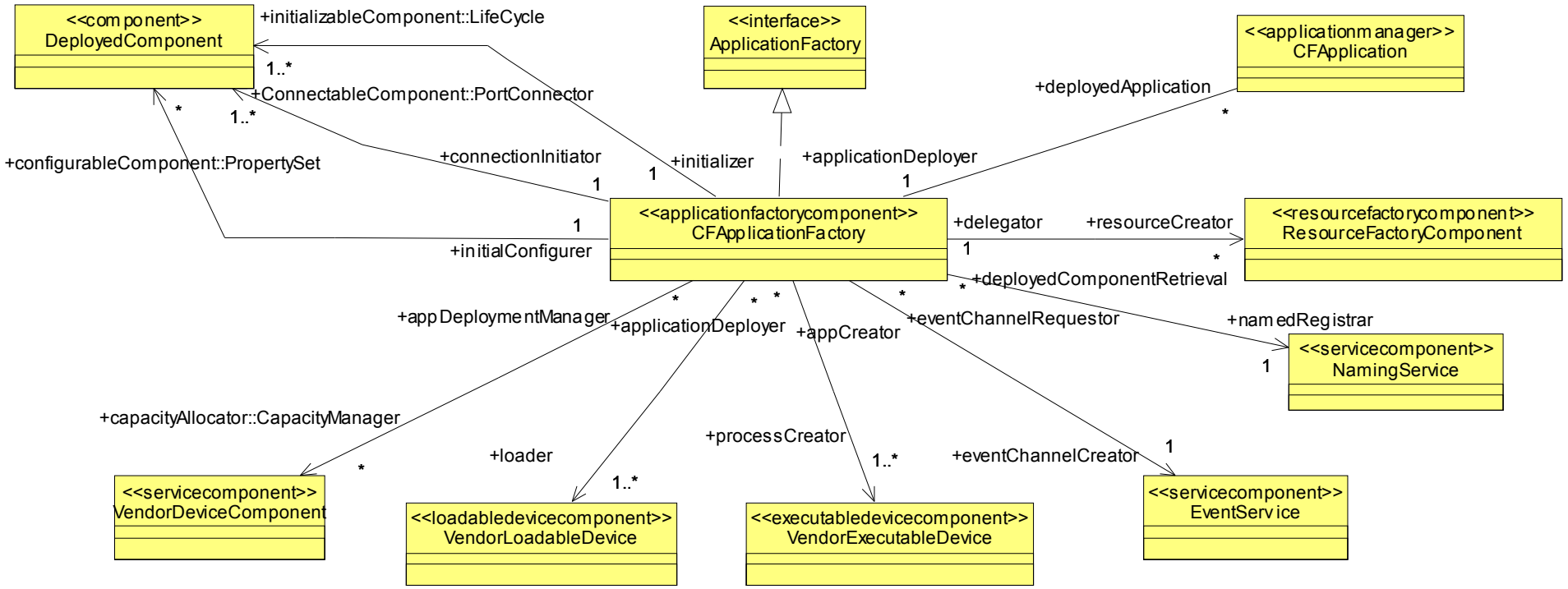
- CF Application Factory Component
  - Comes into existence upon successful installation of an application descriptor (SAD).
  - Creates CF Application Component(s) based upon the associated application descriptor information.
- CF Application Manager Component
  - Proxy for the deployed application assembly
  - Responsible for tear down of the deployed application assembly
- These Components are not developed by application developers but by CF or Platform developers.

CF Application Deployment – Application Factory

- CF Application Factory Component
  - Realizes the CF ApplicationFactory interface.
  - Deployment of application components by application's descriptor
  - Can make capabilities decisions or delegate to device components.
  - Can be directed what devices to use or makes its own decision based upon application deployment requirements

<<interface>> ApplicationFactory
<<readonly>> name : String <<readonly>> softwareProfile : String
create(name : in String, initConfiguration : in Properties, deviceAssignments : in DeviceAssignmentSequence) : ApplicationManager

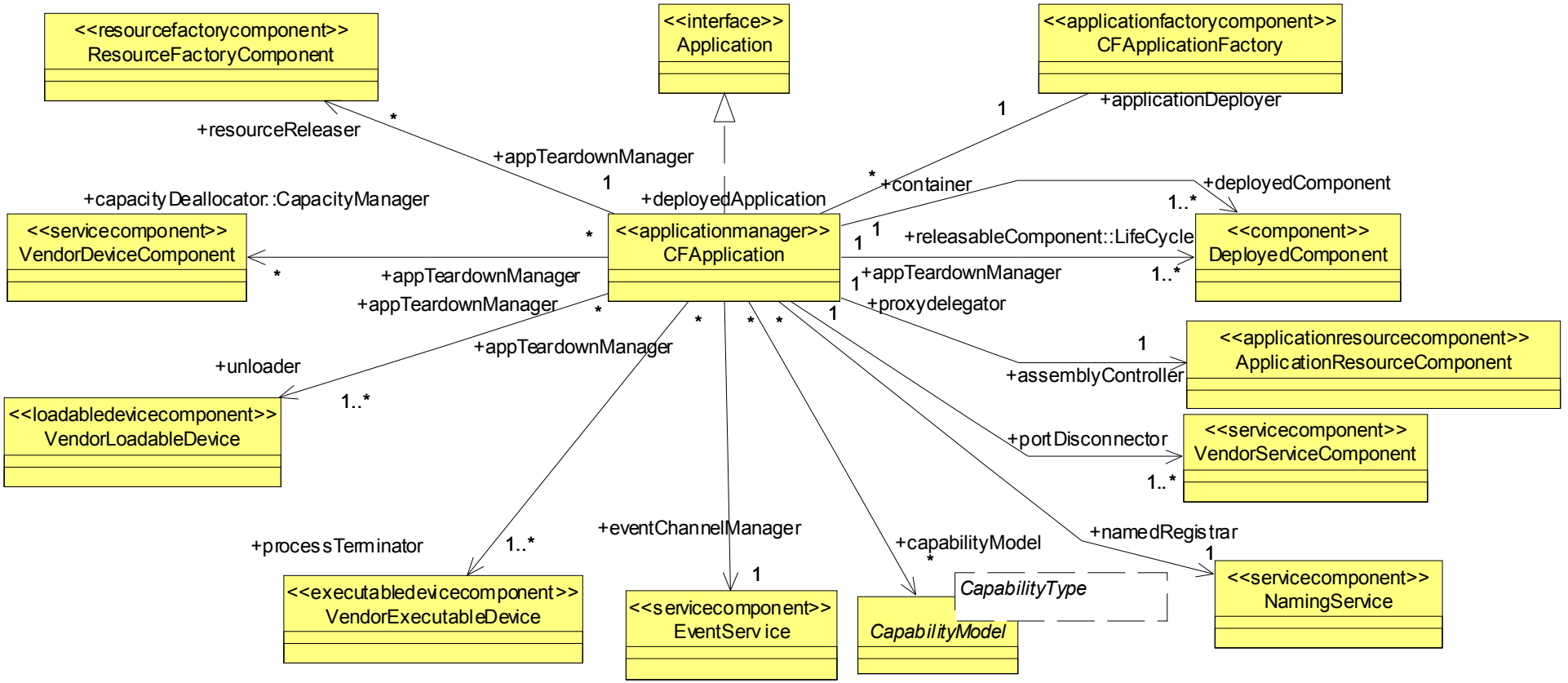
CF Application Deployment – Application Factory Component Illustration



CF Application Deployment – CF Application Manager

- CF Application Manager Component
  - Realizes the CF Application interface.
  - Proxy for the deployed application
  - Provides basic operations for managing the deployed application assembly.
    - Delegates CF Resource operations to Application's component internal assembly controller Resource component
    - Can access deployed application components ports

CF Application Deployment – CF Application Illustration



PRISMTECH
Productivity Tools & Middleware

PRISMTECH
Productivity Tools & Middleware

Software Radio Specification

Component Framework (CF) Profile

CF Infrastructure Service Components

Spectra Software Defined Radio Products

CF Infrastructure Service Components

- Service Components
- Device Components
- File Services

Service Component

- A Service Component that offers service(s) within the environment, which can be used by components.
  - A Service Component is known by its service properties that describe the type and/or capability of the service being offered.
- Managed Service Component is a specialization a Service component that has state behavior.
  - States: Admin, Operational, and Usage.
  - Realizes the StateManagement Interface.
    - Supports the admin, operational, and usage states management.
  - Optional Status Properties

Managed Service Component States

- Operational State
  - ENABLED - the component is functional
  - DISABLED - the component is non-functional
- Usage State is based upon a capacity model
  - Idle – no capacities have been allocated or consumed
  - Active – component in use, with capacity remaining for allocation,
  - Busy – component in use and all capacities have been allocated or consumed
- Admin State
  - LOCKED - The component is reserved for administrative usage only. For example, no capacity requests are granted by a component such as device. Transitions to this state may originate from either the SHUTTING_DOWN or UNLOCKED state.
  - SHUTTING_DOWN - a transition state from UNLOCKED to LOCKED, this state does not allow capacity requests to be granted successfully and is maintained until all capacities are deallocated and the component is not in use.
  - UNLOCKED - The component is available for full usage providing the operationalState is ENABLED. A state transition from SHUTTING_DOWN or LOCKED can occur.

Managed Service Component Optional Status Properties

- ISO/IEC International Standard 10164-2 Status Properties
  - Alarm
  - Availability
  - Control
  - Procedural
  - Standby
  - Unknown
- Not all status properties are applicable for all Device Components.
  - Not all of values are applicable to every class of Device Component.
  - When the value of a status attribute is empty set, implies that none of the status conditions described below are present

Managed Service Component Optional Status Properties

- AlarmStatus
  - configure simple property of ushort type
    - Multi-value, zero or more of the values
    - repair = 2, critical = 4, major = 8, minor = 16, and alarm outstanding = 32
- AvailabilityStatus
  - Query simple property of ushort type
    - Multi-value, zero or more of the values
    - test = 2, failed = 4, power off = 8, off line = 16, off duty = 32, dependency = 64, degraded = 128, not installed = 256, and log full = 512.
- ControlStatus
  - configure simple property of ushort type
    - Multi-value, zero or more of the values
    - subject to test =2, part of services locked = 4, reserved for test = 8, and suspended = 16.

Managed Service Component Optional Status Properties

- StandbyStatus
  - Query simple property of ushort type
    - attribute is single-valued
    - standby hot = 2, standby cold = 4, and providing service = 8.
  - Only meaningful when the back-up relationship role exists.
- ProceduralStatus
  - The proceduralStatus attribute is used to indicate a procedure status for a device component that progresses through a sequence of phases.
  - Query simple property of ushort type
    - initialized required= 2, not initialized = 4, initializing = 8, reporting = 16, and terminating = 32.
- UnknownStatus
  - The unknownStatus attribute is used to indicate that the state of the resource represented by the device component is unknown.
  - Query Simple property of boolean type
    - A value of True means the value of the state attributes may not reflect the actual state of the resource.

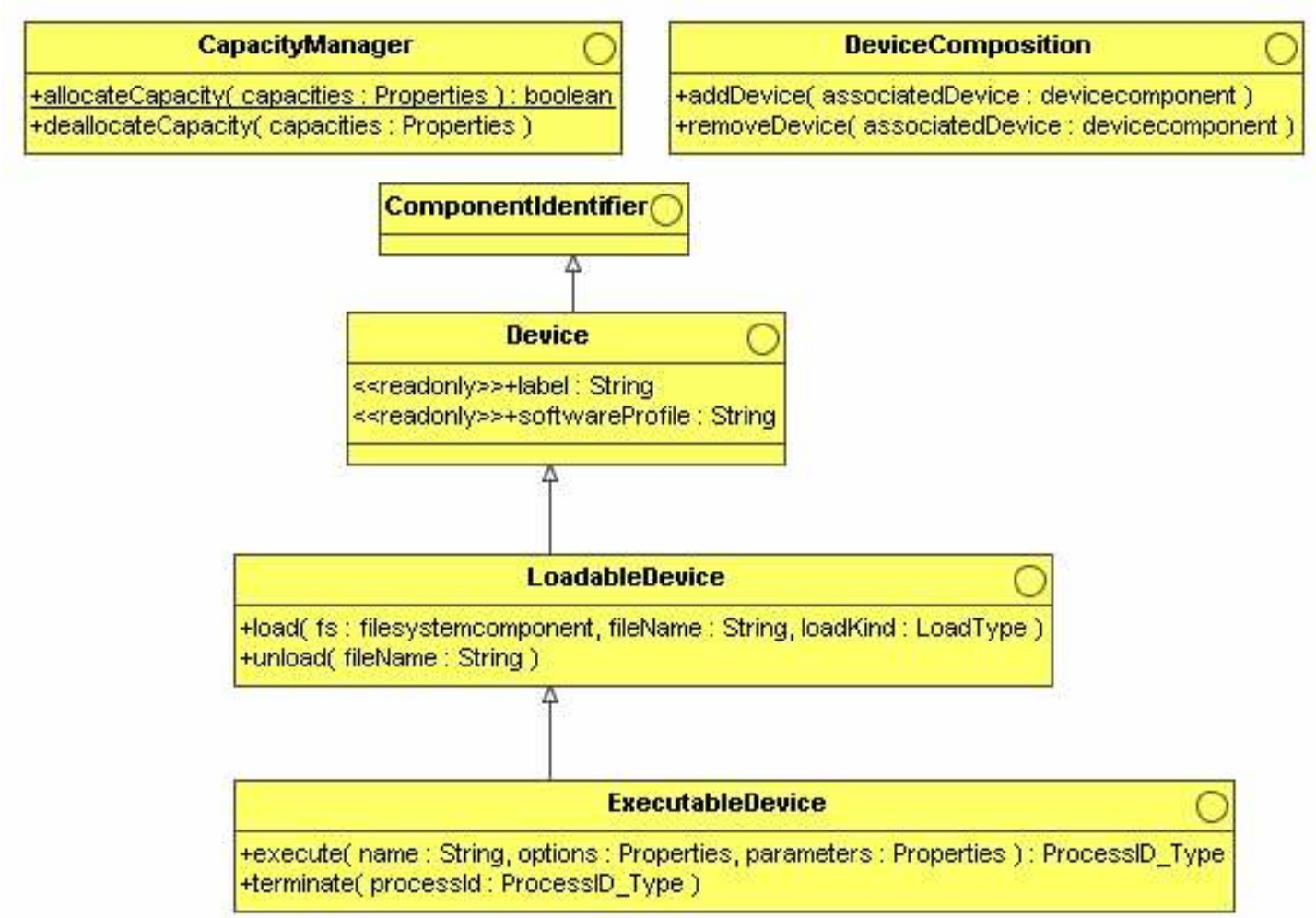
Device Components

- A Device Component is a specialization of Service Component and Component
- A logical device that manages communication equipment
  - A type of service component that has service properties (capabilities):
    - Characteristics – properties that described qualities and/or type of a device. They are usually static in nature.
    - Capacities – computational property values
    - A Device Component can optional manage its service properties, not mandatory.
  - Provides the mechanism (device driver) to directly interface with the communication equipment
  - Can be a Managed Service Component
  - The device components can be extended by the interfaces it provides and/or uses.

Device Component Interfaces

- A set of ***minimal*** generic interfaces for device management
  - CapacityManager – support the management of device's capacities
  - DeviceComposition –supports management of composite devices
  - Device – extends Component interface, an interface that identifies a device component.
  - LoadableDevice – extends Device interface with generic load and unload of software for a communication equipment.
  - ExecutableDevice – extends LoadableDevice interface with generic execution and termination of component containers (e.g., processes, threads, environments).

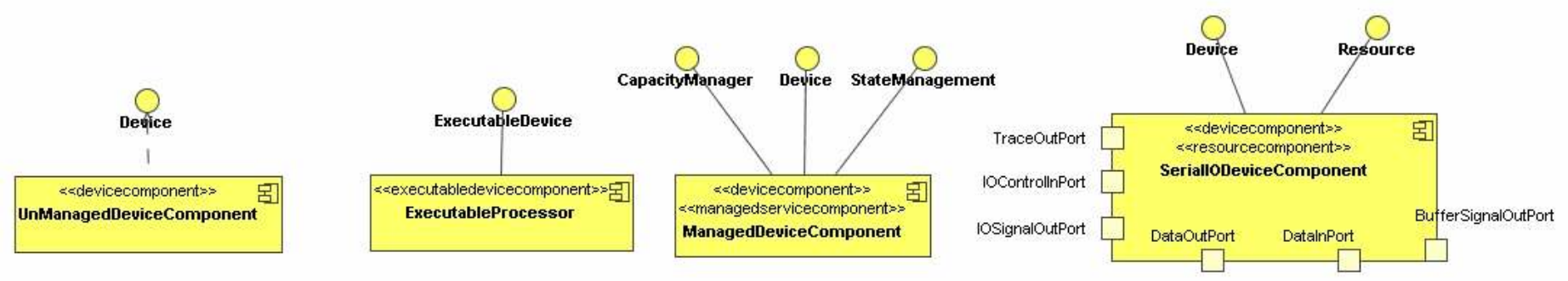
Device Component Interfaces



Low Profile Device Component Definitions

- The OMG UML Profile for SW Radio does not restrict one from building lighter weight SW Radio Device components since a Device Component does not have to manage its Service Properties.
  - Device Components that manage some or all of their Service Properties would require the capacity operations and StateManagement interfaces.

Low Profile Device Component Illustrations



LoadableDevice Component

- Is a component that realizes the LoadableDevice interface
- How a Device performs a load and unload is implementation dependent
- A Loadable Device Component has a LoadKind characteristic property that describes the types of loads supported
- Load operation
  - Performs load based upon the requested load kind and file
  - Should be conformant based upon its LoadKind characteristic property.
  - Multiple loads of the same input fileName do not result in an exception or a duplicate
    - should account for this attempt so that the unload operation behavior can be performed.
  - Raises exceptions when load is not successful
    - Invalid load kind request.
    - Invalid file input
    - Invalid state, not operational or admin locked.
    - Unable to perform request due to system
      - *POSIX error code returned*

LoadableDevice Component

- Unload operation
  - Performs unload based upon the requested input file name parameter and when the number of unload requests matches the number of load requests for the input file name.
  - Raises exceptions when unload is unsuccessful
    - Invalid file input
    - Invalid state, not operational or admin locked

ExecutableDevice Component

- Is a component that realizes the ExecutableDevice interface.
- How a Device performs a Execute and Terminate is implementation dependent
- Execute Operation
  - Creates a main process, thread, and runtime environments to support execution of the requested file/function
    - Stack size and process priority options
    - User defined parameters are passed to user process or function
      - *In the format of argv*
  - Raises exceptions
    - Invalid File Name
    - Invalid Function Name
    - Invalid Parameters
    - Invalid Options
    - Execute System failure
      - *POSIX error code returned*

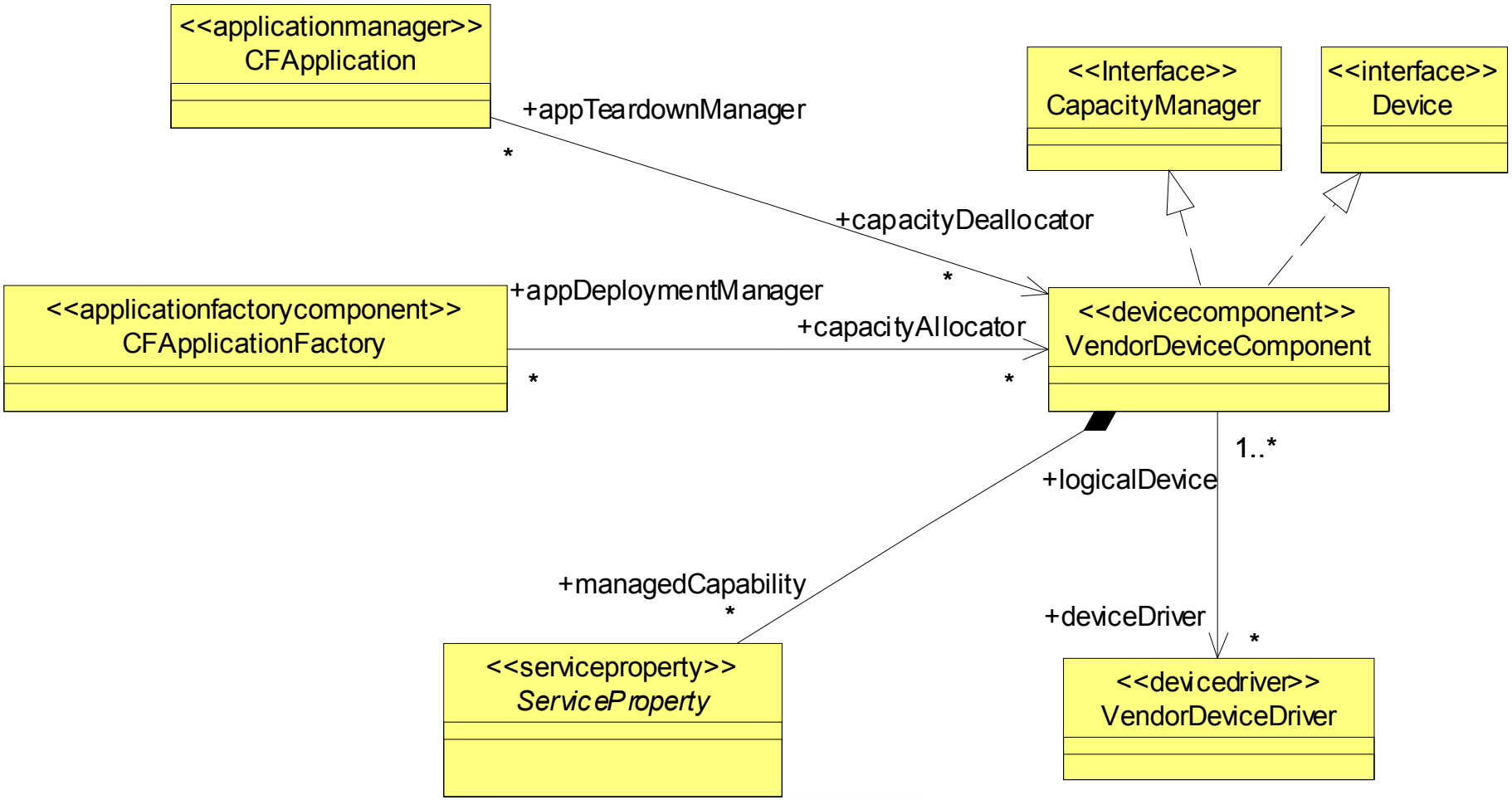
ExecutableDevice Component

- Terminate Operation
  - Terminates the process or thread.
  - Raises exceptions
    - Invalid Process Id
    - Invalid State

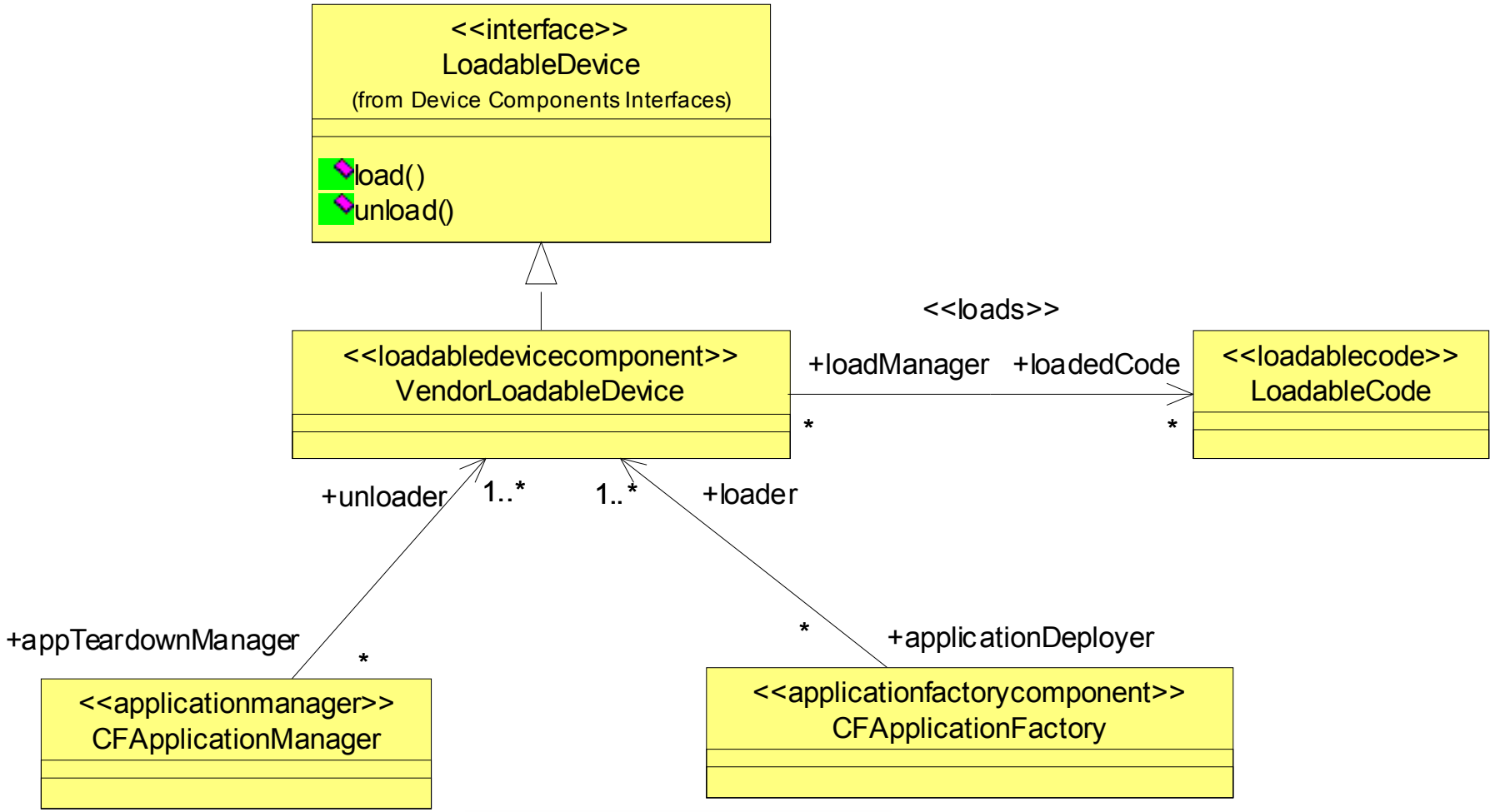
Device Component Kind Illustrations

- Device Component - specialization of Resource Component that realizes the Device interface and is mainly used for managing non-programmable devices such as Input and Output devices.
  - Serial
  - Audio
- Loadable Device Component – specialization of Device Component that realizes the LoadableDevice interface.
  - Transceiver
    - FPGA
- Executable Device Component – specialization of Loadable Device Component that realizes the ExecutableDevice interface.
  - Classified Processor / Link Layer Processor
    - GPP
    - DSP
- DeviceDriver – a component that interfaces with the Radio communication equipment.

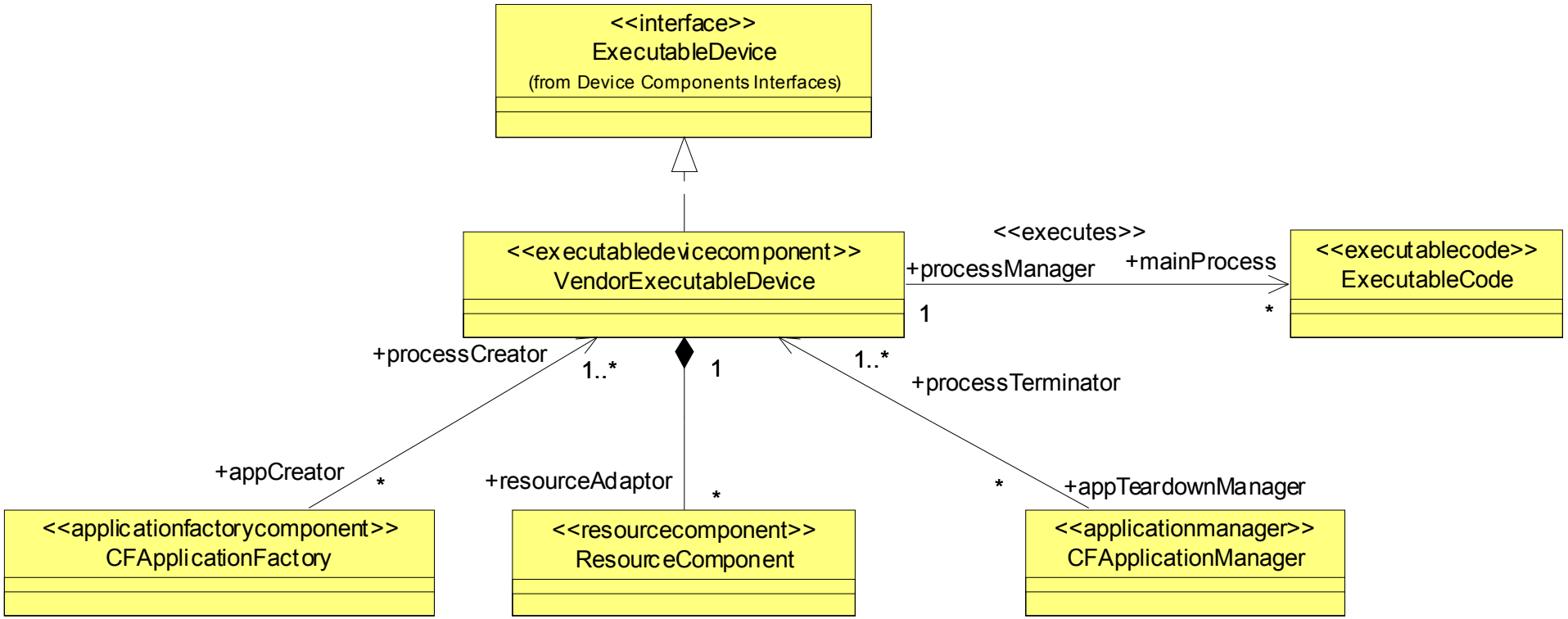
Device Component Illustration



Loadable Device Component Illustration



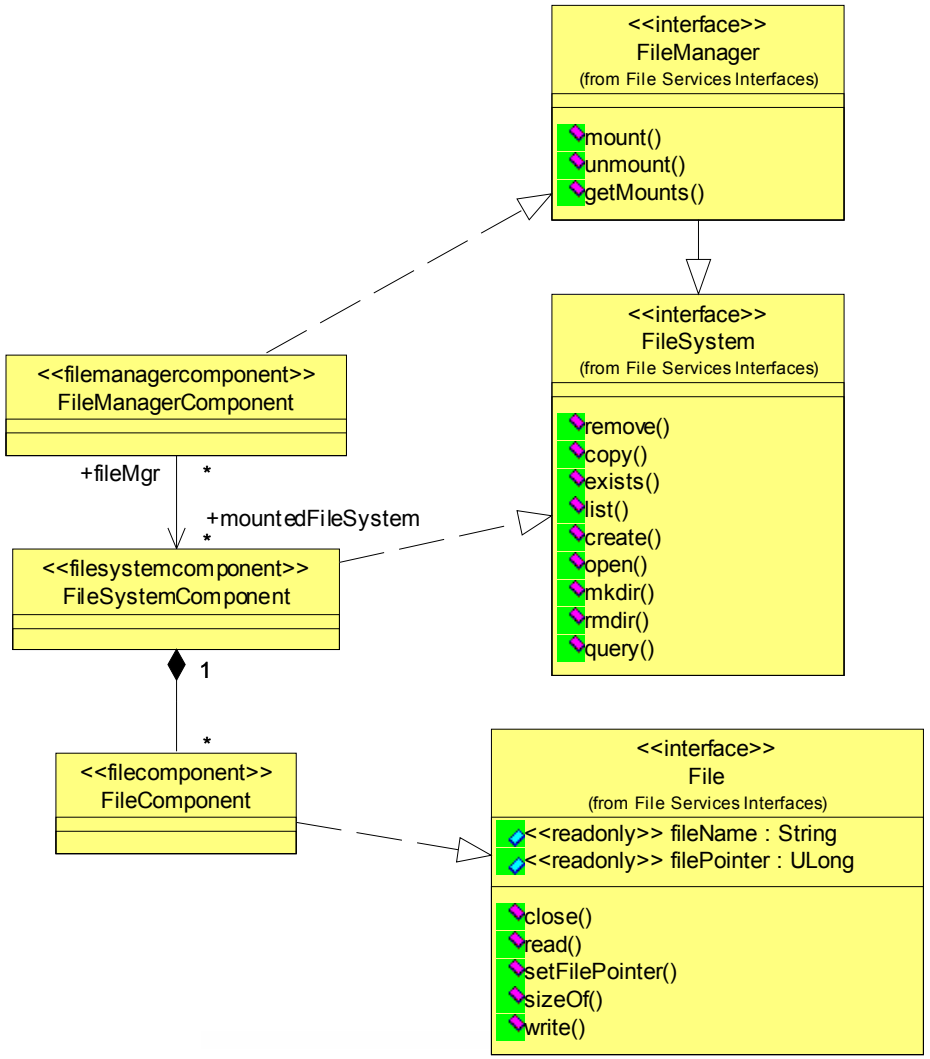
Executable Device Component Illustration



CF Infrastructure Services - File Services

- File Services provide access to the Files and File Systems within a Radio Set.
- Three Interface Types and Capabilities
  - FileManager
    - Manages multiple distributed File Systems
    - Looks like a File System to the client
    - At the Domain Manger Component level and can be at the Device Manager Component level
  - FileSystem
    - Enable remote access to physical file systems
    - Allows creation, deletion, copying, etc. of files
  - File
    - Provides access to files within the radio
    - Allows access across processor boundaries (distributed File Systems)

CF Infrastructure Services – File Services Illustration



PRISMTECH
Productivity Tools & Middleware

Software Radio Specification

Communication Channel Profile

Spectra Software Defined Radio Products

Communication Channel Profile

- A language to describe a specific hardware platform upon which applications execute.
- Enables the deployment and configuration machinery to acquire knowledge about the platform capabilities.
- Expressed as a UML Profile that are extensions of the Class, Device, Port, and Property elements.
- Definition can be captured in XMI for interoperability with other tools.
- Definition can be transformed into any technology such as XML.

UML Profile for Communication Channel

- Communication Equipment – concepts that are required to model communication equipment .
- Communication Channel – concepts that are required to model the communication equipment that make up a Radio Set's channel and defines different parts of channel (security, I/O, processing, physical).
- Radio Management – concepts that are required for managing a Radio Set and its channels.
- Communication Channel Facilities – interfaces and components for physical layer and radio set control.

Communication Channel Profile - Communication Equipment

- CommEquipment in the profile is the overall base device for describing features and constraints that are common to all radio devices.
  - Equipment Size
  - Equipment Weight
  - Min and Max Operating Temperatures
  - Mean Time Between Failure
  - Power Consumption
  - Radiation Capability
  - Last Maintenance Check
  - Maintenance Period
  - Temperature Status
- Extension of UML Device
- Communication Equipment Port Types
  - Analog & Digital
- Communication Equipment Property Types
  - Characteristic, Capacity, Configure, and Query

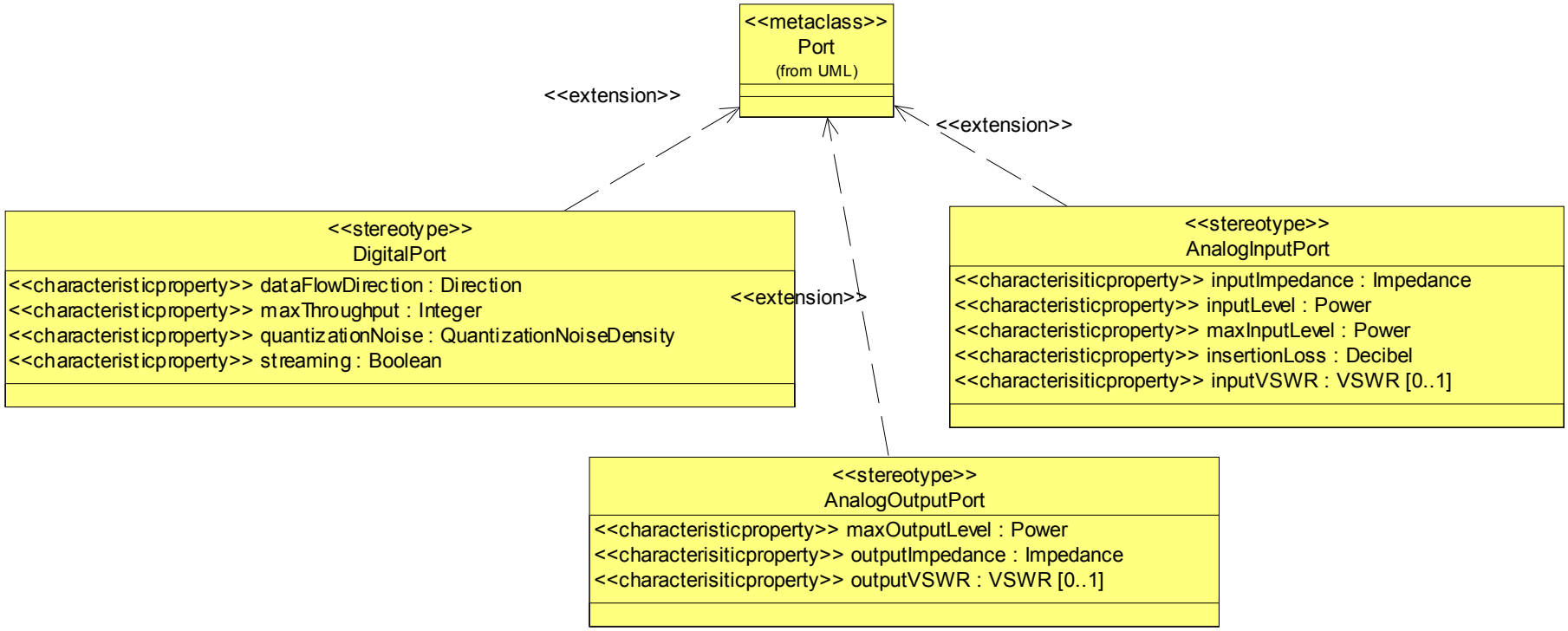
Communication Equipment - Ports

- Extension of UML Port
  - Analog Input Port receives an analog signal and properties are:
    - Input Impedance
    - Current Input Power Level
    - Insertion Loss
    - Input Voltage Standing Wave Ratio (VSWR) of the port.
    - Maximum Input Power Level
  - Analog Output Port transmits an analog signal and properties are:
    - Output Impedance
    - Output Voltage Standing Wave Ratio (VSWR) of the port.
    - Maximum Output Power Level

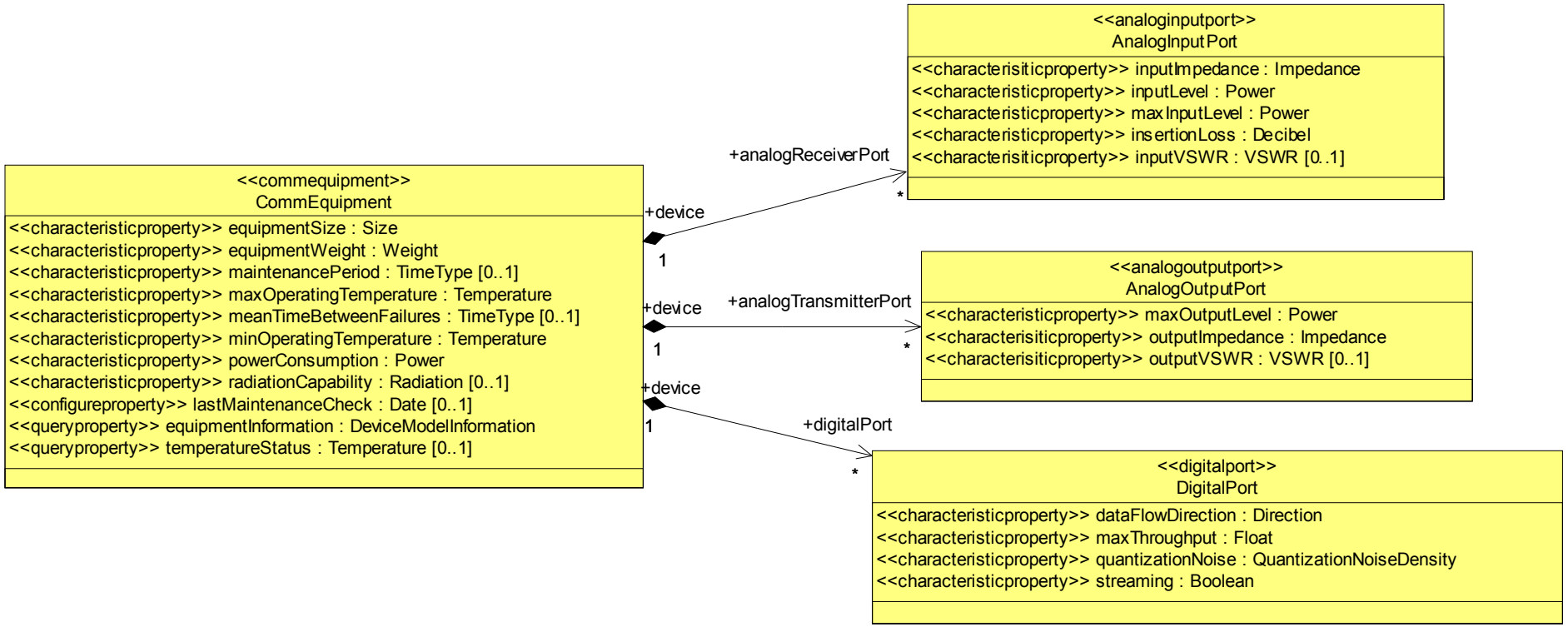
Communication Equipment - Ports

- Extension of UML Port
  - Digital Port transmits or receives a digital signal and properties are:
    - Data Flow Direction
    - Maximum Throughput
    - Quantization Noise
    - Streaming Port

Communication Equipment – Ports



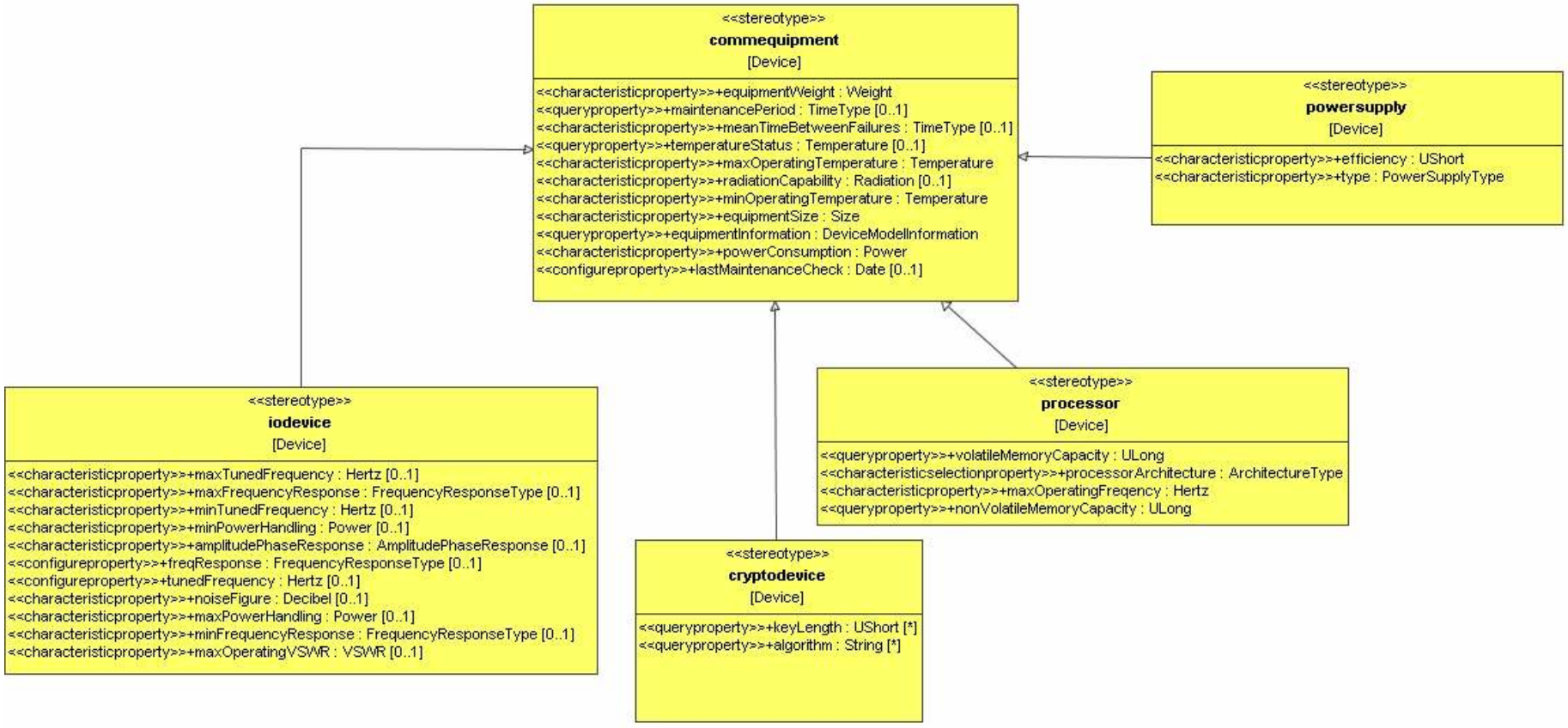
Communication Equipment M1 Illustration



Communication Equipment Specializations

- Communication Equipment Specializations:
  - Crypto Device - performs encryption and decryption on asset of data.
  - I/O Device - describes the constraints and attributes that are appropriate for I/O devices
    - Antenna, Amplifier, Audio, Filter, Frequency Converter, etc.
  - Power Supply - provides electrical power to other devices.
  - Processor - processes digital or analog data.

Communication Equipment Specializations



Communication Equipment – I/O Device Properties

- I/O Device properties in addition to the CommEquipment properties
  - Min and Max Power Handling
  - Min Power Handling
  - Noise Figure
  - Max Operating VSWR
  - Freq Response
  - Tuned Frequency
  - Min and Max Frequency Response
  - Min and Max Frequency
  - Amplitude Phase Response

Communication Equipment – IODevice Specializations and Their Properties

- Amplifier
  - Duty Cycle
  - Gain
  - Min and Max Gain
- Antenna
  - Calibration
  - Radiation Pattern
  - polarization
  - Type
  - Min and Max Radiation Pattern
  - Polarization Capability
- Audio Device

Communication Equipment – IODevice Specializations and Their Properties

- Digital Converter
  - Sample Rate
  - Min and max Sample Rate
  - Sample Size
  - Phase Noise
- Filter

Communication Equipment – IODevice Specializations and Their Properties

- Frequency Converter
  - Current Input and Output Frequency
  - Min and Max Input Frequency
  - Min and Max Output Frequency
  - Lo Input and Output Leakage Power
  - Output To Input Leakage
  - Phase Noise
  - Lo Stability
    - Hopping Frequency Converter
      - *Next Input Frequency*
      - *Next Output Frequency*
- Microphone
- Serial IO

Communication Equipment – IODevice Specializations and Their Properties

- Radiating Element
  - Active,
  - Radiation Pattern
  - Polarization
  - Type
  - Position In Antenna Array
- Switch
  - Input Output Isolation
  - Switch Setting

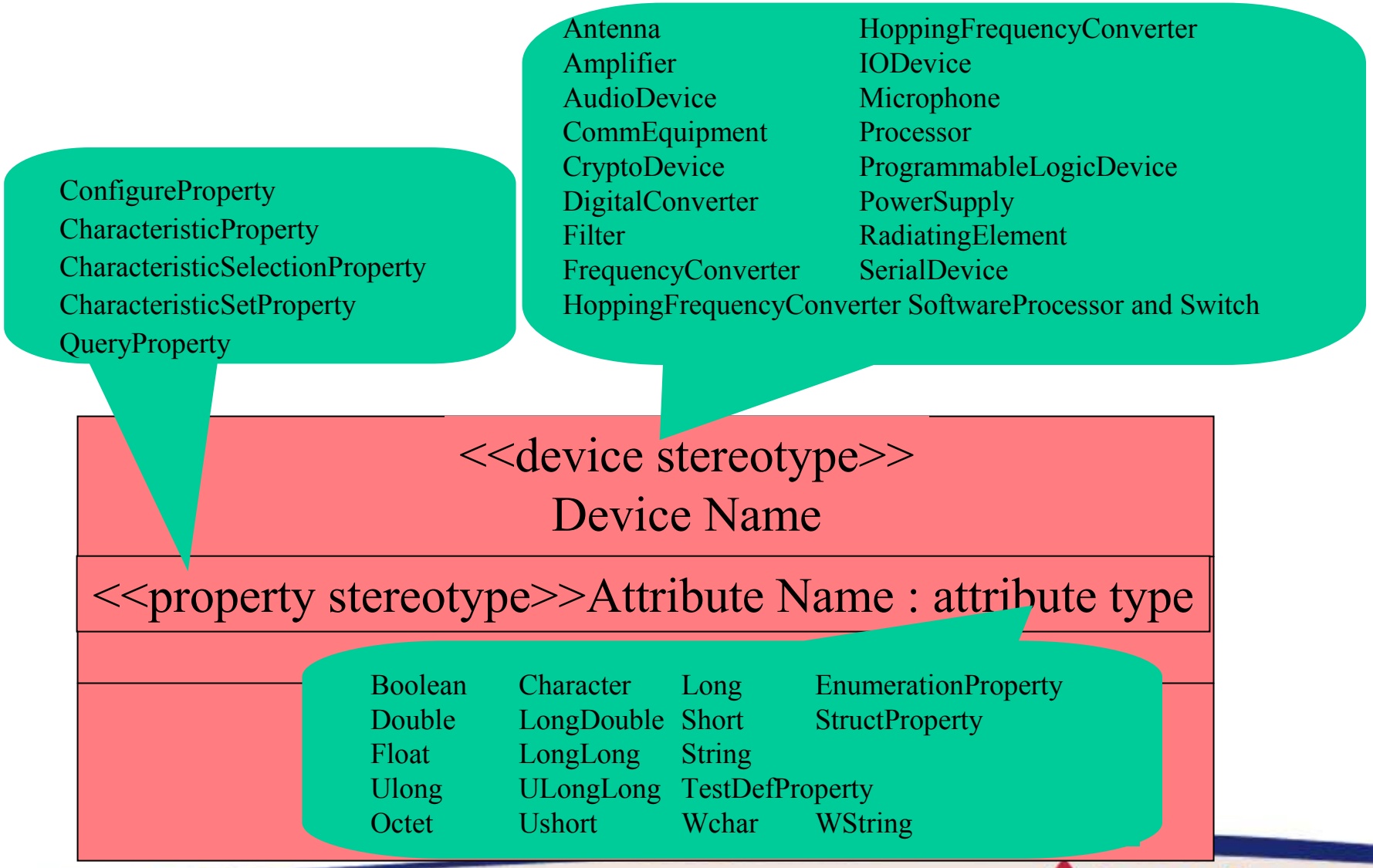
Communication Equipment – Power Supply

- A device that provides electrical power to Communication Equipment components.
  - Efficiency
  - Power Supply Type

Communication Equipment – Processor Specializations and Their Properties

- Processor
  - Processor Architecture
  - Max Operating Frequency
  - Non Volatile Memory
  - Capacity
  - Volatile Memory Capacity
- Processor Specializations
  - Programmable Logic Device
    - Logic Unit Capacity
    - Reconfigurability
    - Time For Reconfiguration
  - Software Processor
    - Operating Environment

Communication Equipment DSL



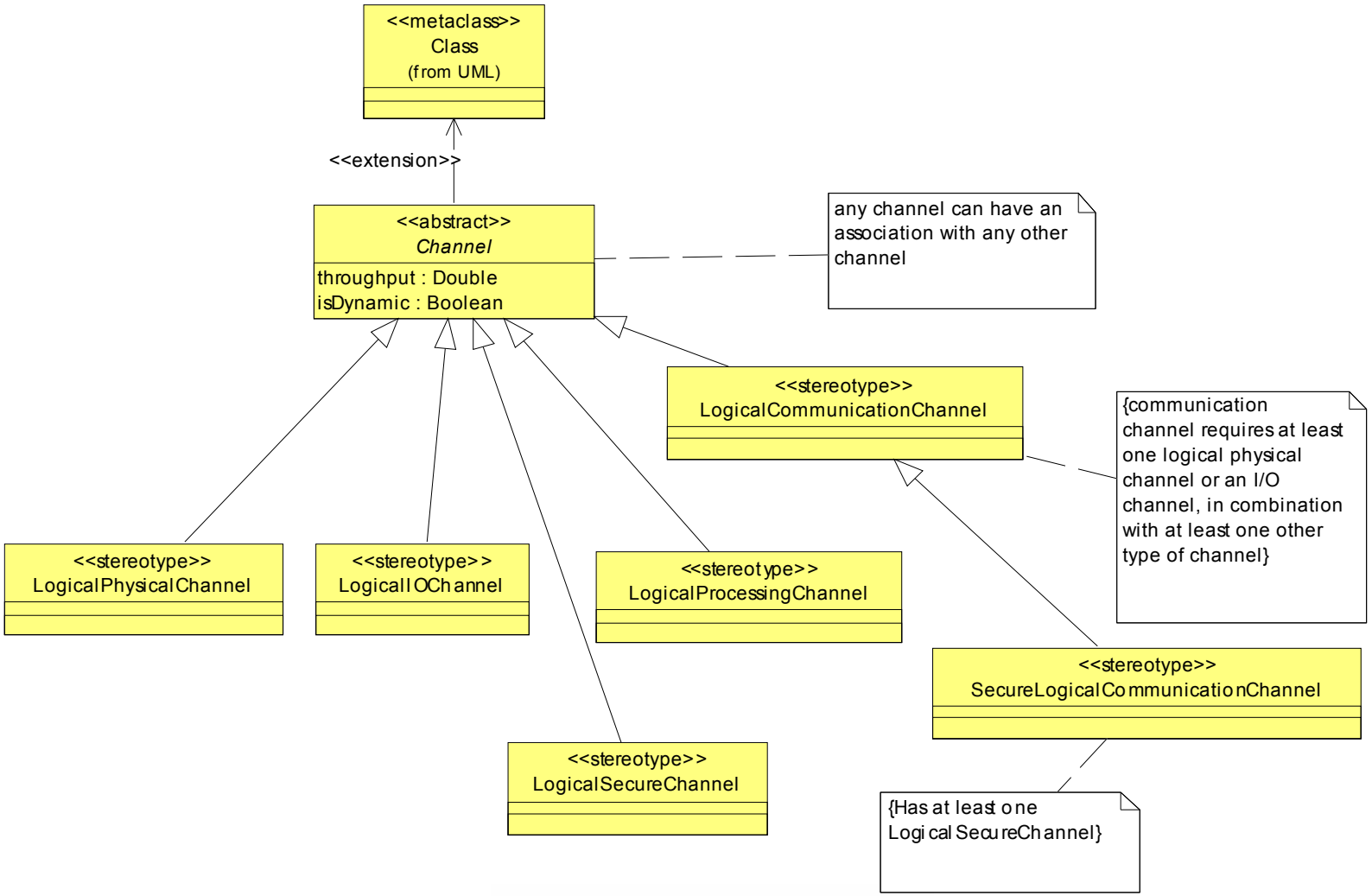
Communication Channel Profile – Communication Channel

- Data description for the collection and interconnection of the radio's communication equipment necessary for a particular waveform application to be able to provide communication.
- Channel concept is an extension of UML Class that defines the relationships and attributes for channel types.
  - Is associated with one to many Communication Equipments. The types of Communication Equipments associated with a channel depends on the type of channel.

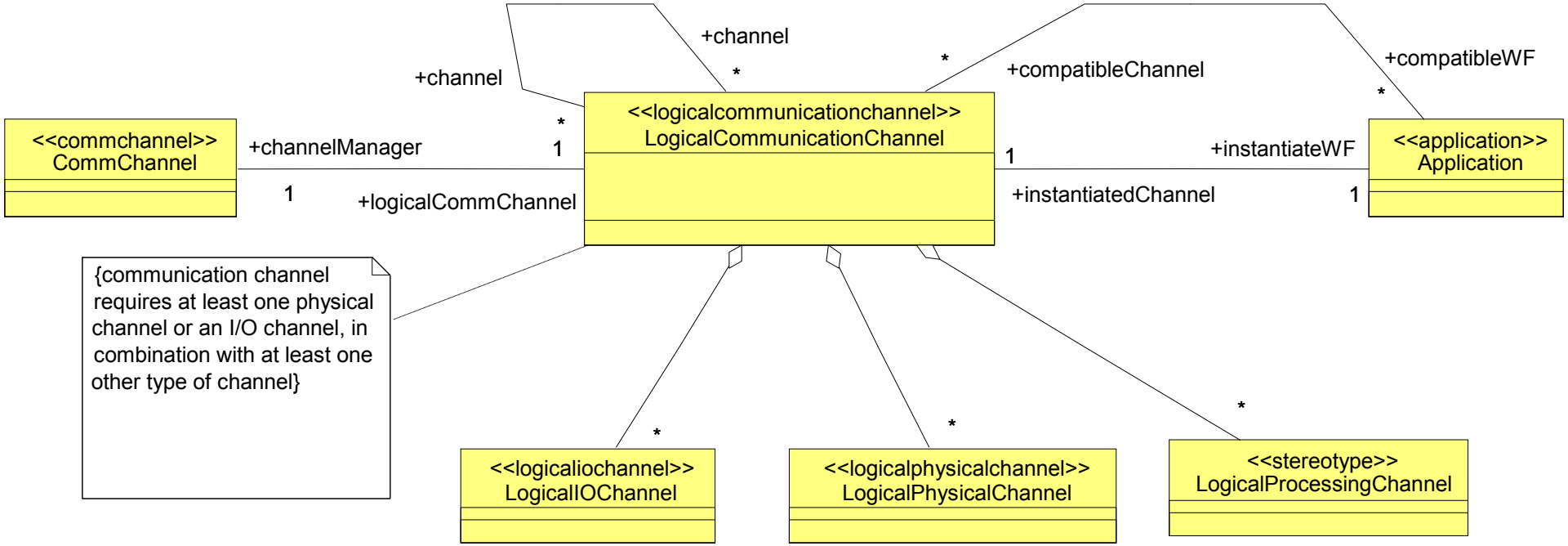
Communication Channel – Channel Stereotype Specializations

- Logical Communication Channel is an aggregate of Logical Processing Channel, Logical IO Channel and Logical Physical channel.
- Logical Physical Channel stereotype consist of all devices processing the analog signal after digitization, to and including the antenna(s).
- Logical Processing Channel provides the processing nodes for waveform applications and radio services used by the waveforms running on the processing channel's operating environment(s).
- Logical IO Channel provides for the base-band connection to the radio and consists of the devices that format, encode, decode, etc.
- Logical Security Channel provides the processing node(s) for security applications applicable to communications.
- Secure Logical Communication Channel specialization of Logical Communication Channel that adds aggregation relationship to the Logical Security Channel.

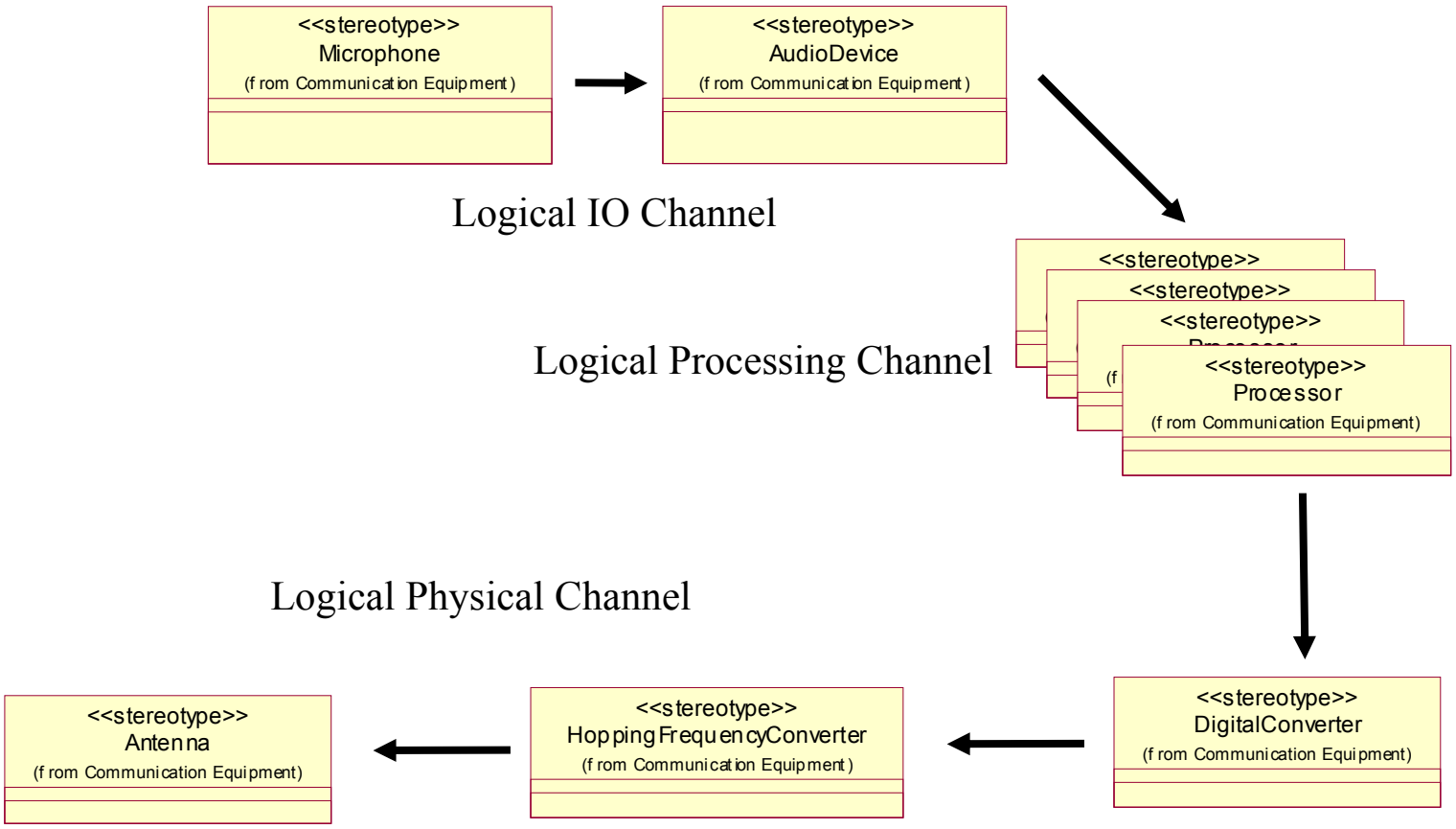
Communication Channel – Channel Stereotype Specializations



Communication Channel – Logical Communication Channel Illustration



Logical Communication Channel Illustration



Communication Channel Profile - Radio Management

- Comm Channel Component - associated with static or a dynamic LogicalCommunicationChannel. The devices associated with a static LogicalCommunicationChannel do not vary over the life cycle of the communication channel. For dynamic LogicalCommunicationChannel the devices can vary during the life cycle of the communication channel.
- Radio Manager Component - extends the DomainManagerComponent by providing communication channel management within the RadioSet.
- Radio System Manager - is responsible for control and management tasks for a RadioSystem. It may be associated with one or more RadioManagers which are used to control the RadioSets that a RadioSystem consists of.

Communication Channel Profile - Communication Channel Facilities

- Physical Layer Facilities
  - Modem Facilities
    - The modem facilities include all digital signal processing elements required to convert bits into symbols and vice versa.
  - RF/IF Facilities
    - The RF/IF Facilities is used to configure and control the basic devices of the physical channel. The granularity at which these interfaces are implemented is not specified.
  - I/O Facilities - Defines the configuration properties for Audio and Serial Facilities
- Radio Set Facilities
  - Provides interfaces for radio and channel management.

Communication Channel Facilities- Modem Facilities

- PNSequenceGenerator
  - <<configureproperty>>chipRate: Float
  - <<configureproperty>>polynomial: Polynomial [1..*]
  - <<configureproperty>>modulus: UShort
  - <<configureproperty>>seed: ULongLong
  - <<configureproperty>>columns: UShort
  - <<configureproperty>>rows: UShort
- Transform
  - <<configureproperty>>blockSize: ULong
  - <<configureproperty>>transform: TransformType
  - <<configureproperty>>overlap: ULong
- Channel Coding
  - <<configureproperty>>codeRate: Float
- Source Coding
  - <<configureproperty>>codeRate: Float

Communication Channel Facilities – Modem Facilities

- Latency
  - <<queryproperty>>processingLatency: Seconds
- Block Interleaver
  - <<configureproperty>>columns: UShort
  - <<configureproperty>>rows: UShort
- Convolutional Interleaver
  - <<configureproperty>>delays: UShort [1..*]
- Helical Interleaver
  - <<configureproperty>>columns: UShort
  - <<configureproperty>>groupSize: UShort
  - <<configureproperty>>stepSize: UShort
- Mapper
  - <<configureproperty>>baudRate: UShort
  - • <<configureproperty>>constellation: String
  - • <<configureproperty>>bitPatternMapping: BitsToSymbolsMapping [1..*]

Communication Channel Facilities – RF Facilities

- Frequency Response
  - <<configureproperty>>frequencyResponse: FrequencyResponsePoint [1..*]
  - <<configureproperty>>tunedFrequency: Hertz
- Radiation Pattern
  - <<configureproperty>>radiationPattern: RadiationPattern
  - <<configureproperty>>patternOrientation: PatternOrientation
- Polarization
  - <<configureproperty>>orientation: PolarizationKind
  - <<configureproperty>>ellipticity: Float
- Frequency Converter
  - <<configureproperty>>nextInputFrequency: Hertz
  - <<configureproperty>>nextOutputFrequency: Hertz
  - <<configureproperty>>currentInputFrequency: Hertz
  - <<configureproperty>>currentOutputFrequency: Hertz

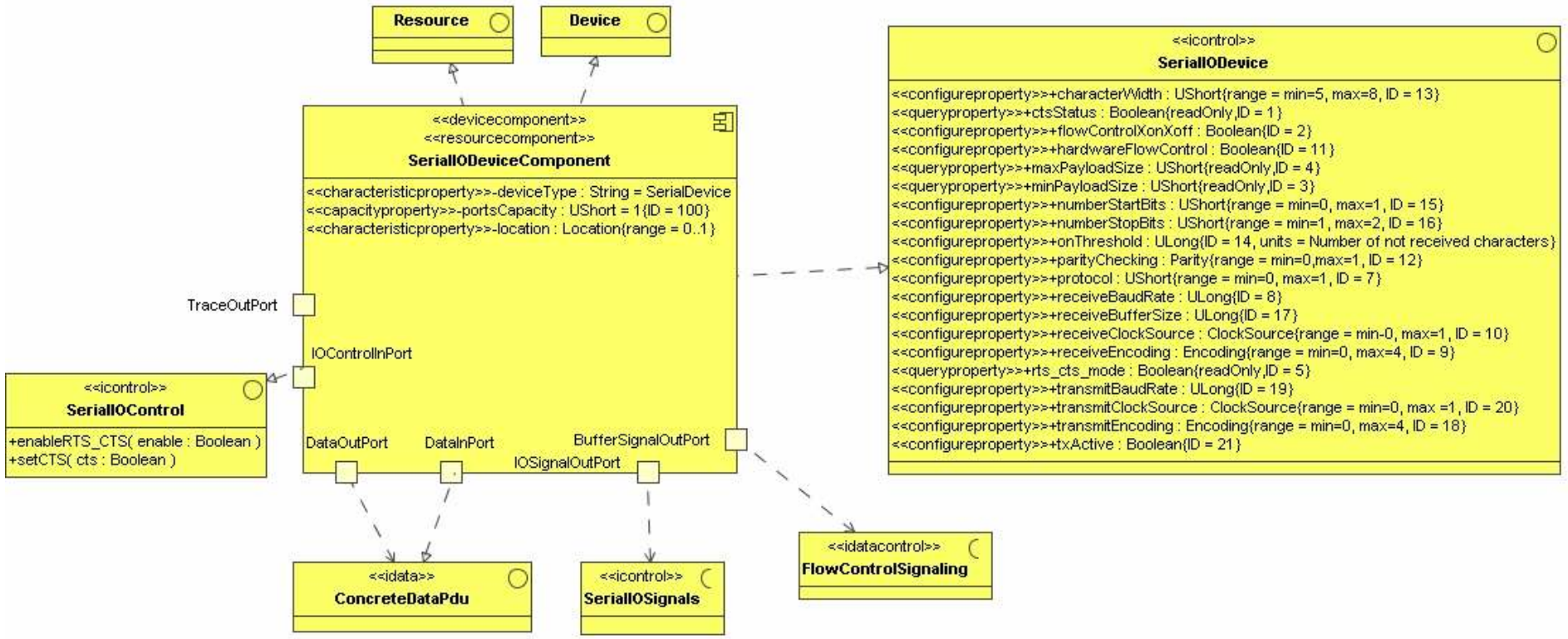
Communication Channel Facilities – RF Facilities

- Sample Rate
  - <<configureproperty>>sampleRate: Hertz
- Average Power
  - <<configureproperty>>averagePower: Power
- Switch
  - <<configureproperty>>numberOfPorts: UShort
  - <<configureproperty>>switchSetting: SwitchMapping [0..*]

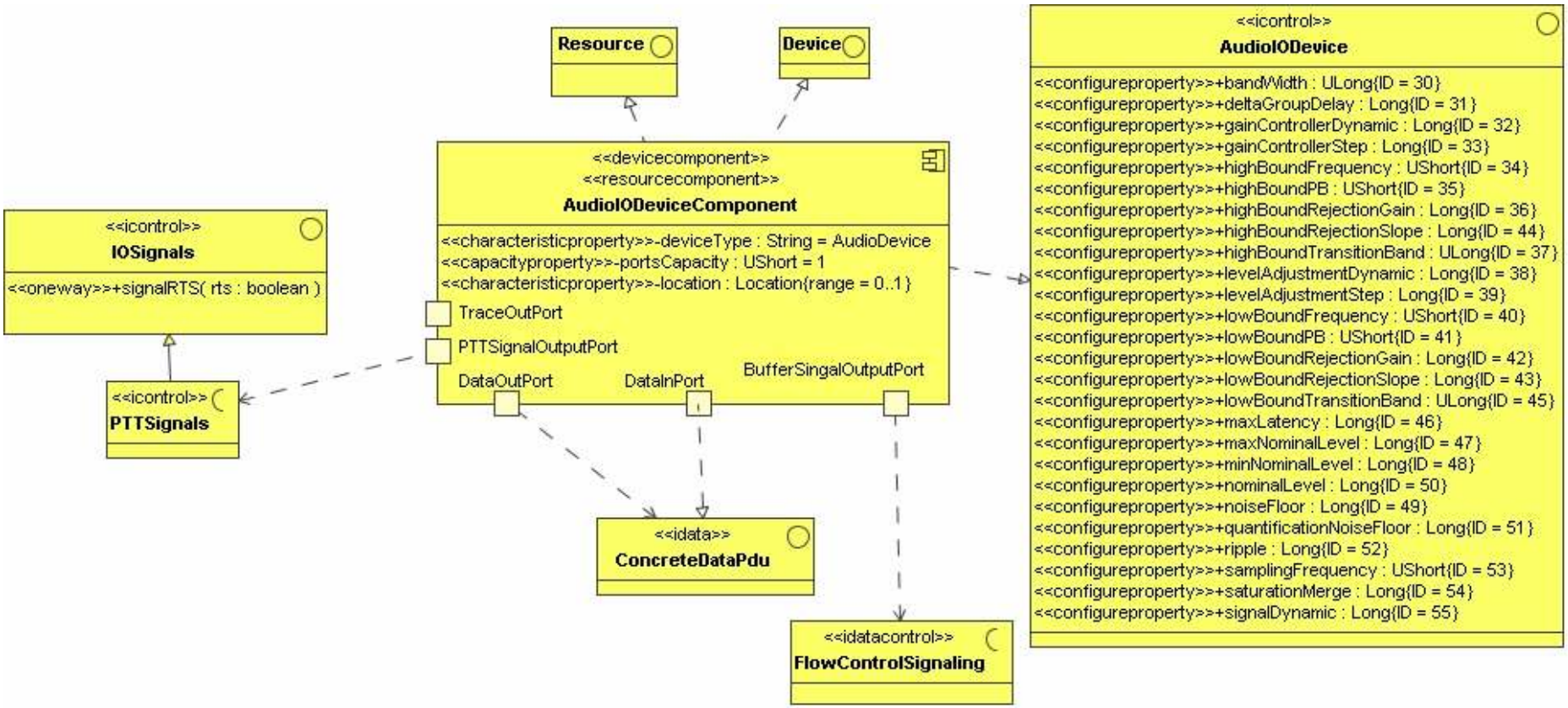
RF Facilities

- OMG Submissions In progress
  - PIM and PSM for Digital Intermediate Frequency Interface
    - http://www.omg.org/cgi-bin/doc?sbc/2004-8-15
  - PIM and PSM for Smart Antenna
    - http://www.omg.org/cgi-bin/doc?sbc/2006-12-10
  - Power Conservation Services
    - http://www.omg.org/cgi-bin/doc?realtime/2004-2-8

IO Facilities – SerialIO Component



IO Facilities – Audio Component



PRISMTECH
Productivity Tools & Middleware

Software Radio Specification

Software Radio Facilities

Spectra Software Defined Radio Products

- Software Radio Facilities
  - Component Framework Facilities PIM
    - *Component Framework Specification Volume*
  - Common and Data Link Layer Facilities PIM
    - ***Common and Data Link Layer Facilities Specification Volume***
  - Communication Channel Facilities PIM
    - *Communication Channel and Equipment Specification Volume*

Common and Data Link Layer Facilities PIM

- Defines waveform related data and control PIM interfaces using the Software Radio Profile that can be used to define a waveform or platform component.
- Based on the “Extended” OSI Model*
- Descriptions for the interfaces and components realizing the interfaces that follows the similar format as Software Radio profiles.
- Product of a survey of existing specs such as: 3GPP, DLPI, GLoMo, OBSAI, CPRI, 802.x, X.200e

*** ISO 7498-1: Open System Interconnection – Basic Reference Model**

Common and Data Link Layer Facilities PIM

- Common Radio Facilities
  - Provides common service definitions that are applicable for all applications (waveforms or radio control)
    - OMG Lightweight Services (log, event, naming, etc.)
- Common Layer Facilities
  - Provides interfaces that cross cut through facilities that correlate to layers. These interfaces can be viewed as building blocks for SWRadio components that realize multiple interfaces.
  - Protocol Data Unit, Error Control, Flow Control, Measurement, Quality of Service, and Stream Facilities

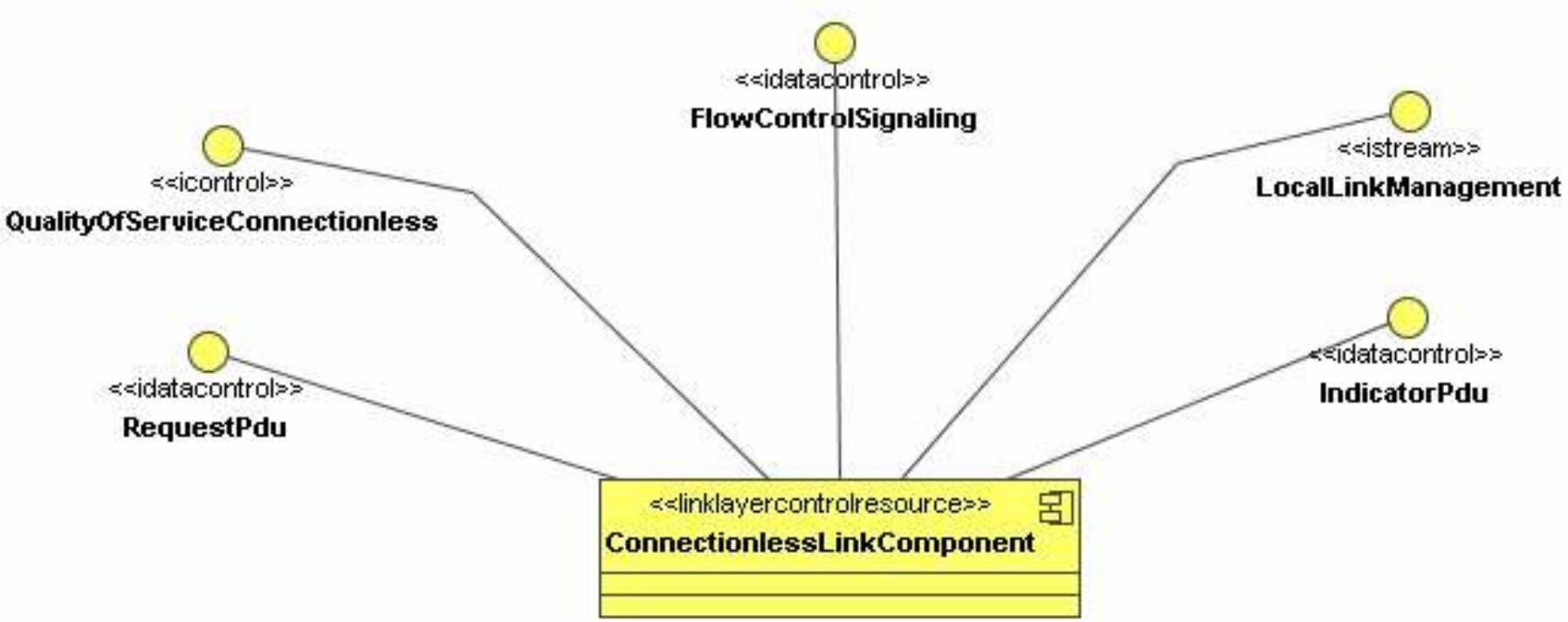
Common and Data Link Layer Facilities PIM – Common Layer Facilities

- Quality of Service Facilities - The set of interfaces that define the quality of service related functionality.
- Flow Control Facilities - The set of interfaces that control communication flow between senders and receivers.
- Measurement Facilities - The set of interfaces that provide measurement parameters and intervals.
- Error Control Facilities - The set of interfaces that allow the Receiver to tell the Sender about frames damaged or lost during transmission, and coordinates the re-transmission of those frames by the Sender.
- PDU Facilities - The set of interfaces that define the Protocol Data Unit (PDU) used in communication among radio sets as well as inter-component communication within a radio.
- Stream Facilities - The set of interfaces that define the stream concept used in communication among radio sets as well as inter-component communication within a radio.

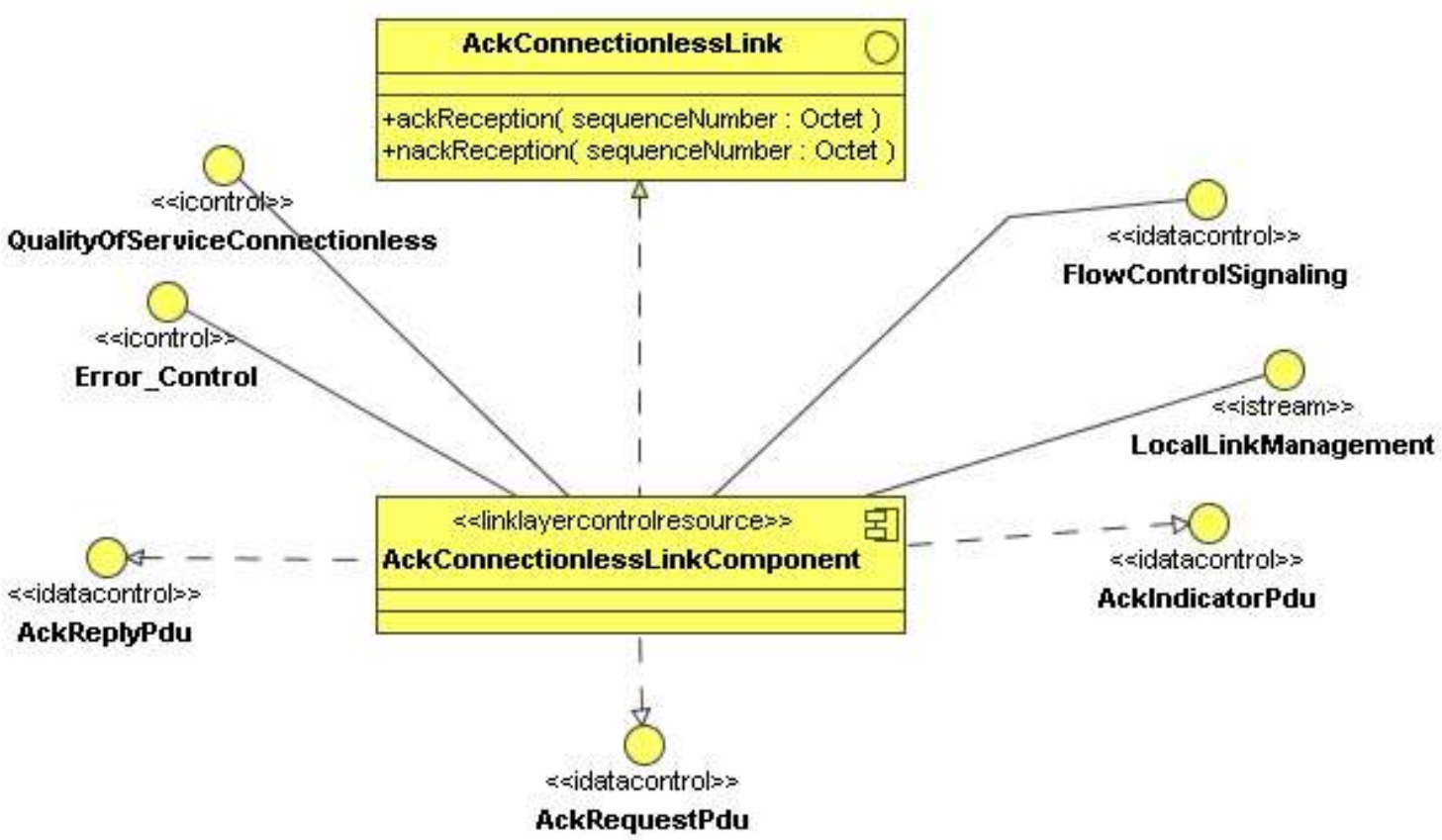
Common and Data Link Layer Facilities PIM, cont'd

- Data Link Facilities
  - Link Layer Control (LLC) facilities. LLC layer provides facilities to upper layers, for management of communication links between two or more radio sets.
  - Data Link Layer (Connectionless, Ack ConnectionLess, Connection), and Medium Access Control Facilities

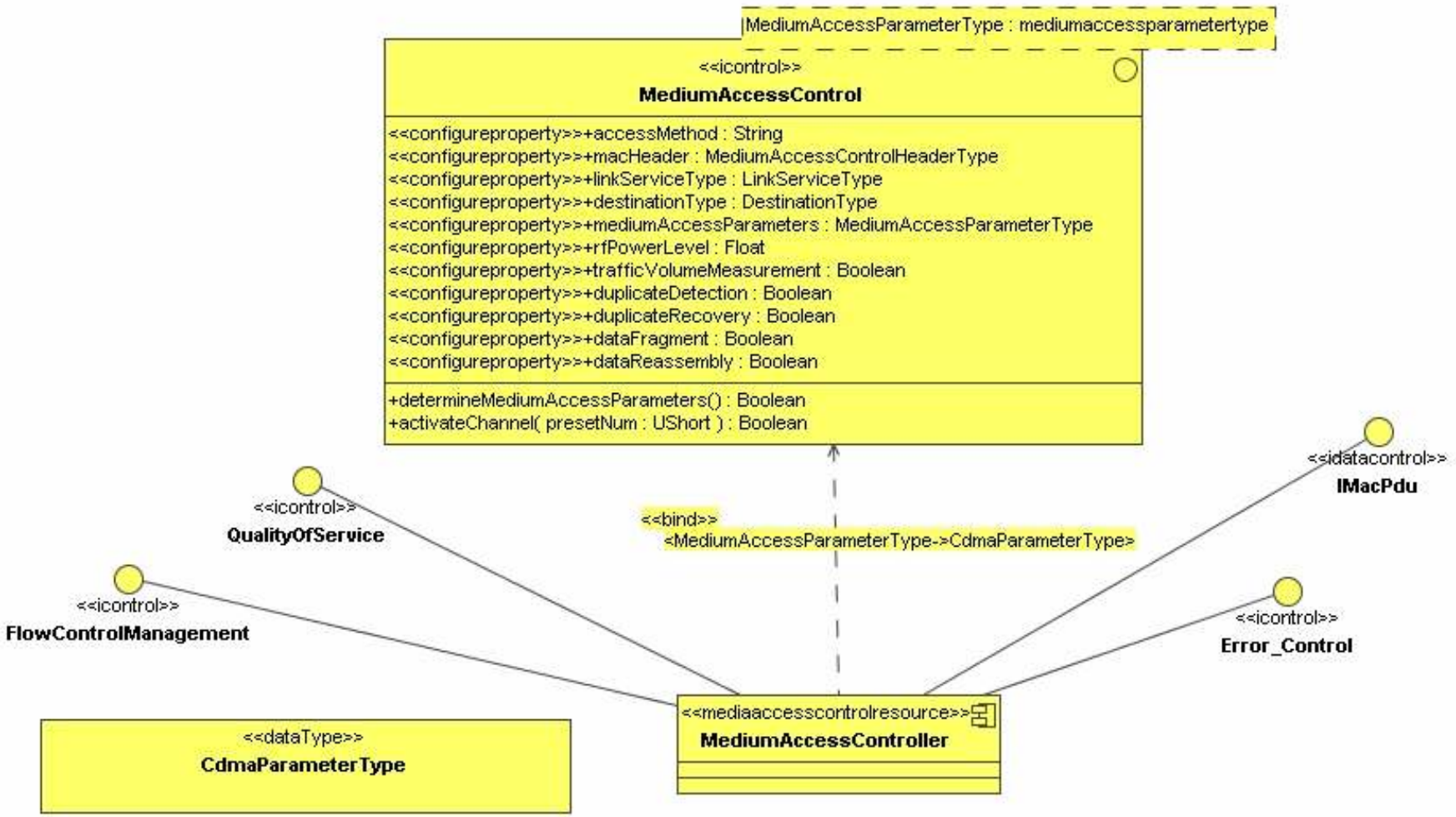
Connectionless Link Component



Acknowledged Connectionless Link Component



Medium Access Controller Illustration



PRISMTECH
Productivity Tools & Middleware

Software Radio Specification

Platform Specific Technologies

Spectra Software Defined Radio Products

Platform Specific Technologies

- At this time only CORBA interfaces, XML descriptors and POSIX profiles are defined
  - These definitions are non-normative.
- The normative is captured in the XMI for the profiles and Software Radio Facilities.
- Transformations rules are specified in the specification for transforming PIM into PSM.
- Other PSMs could be defined along with there transformation rules

Software Radio PSM – CORBA Modules

- UML Interfaces map to CORBA IDL
  - The Component Framework Profile interfaces map to the CF CORBA module
    - CF
      - *StandardEvent, PortTypes, FileServices*
  - The Software Radio PIM Facilities map to the DfSWRadio CORBA module.
    - DfSWRadio
      - *CommonLayer, DataLinkLayer, CommonRadio, PhysicalLayer, RadioControl*
  - CF and DfSWRadio IDL file has been broken apart into multiple files to reduce foot print unneeded client and skeleton code.

Software Radio PSM – CF CORBA IDL Files

- CF CORBA Module IDL Files
  - CF.idl
  - CFApplications.idl
  - CFBaseTypes.idl
  - CFCommonTypes.idl
  - CFDeviceManager.idl
  - CFDeviceManagerRegistration.idl
  - CFDevices.idl
  - CFDomainEventChannels.idl
  - CFDomainInstallation.idl
  - CFDomainManager.idl
  - CFResourceFactory.idl
  - CFResources.idl
  - CFServiceRegistration.idl
  - CFStateManagement.idl
- File Services
  - CFFile.idl
  - CFFileManager.idl
  - CFFileSystem.idl

Software Radio PSM – CF CORBA IDL Files

- CF PortTypes CORBA Module IDL Files
  - CFPortTypes.idl
  - CFPortTypes_BooleanSequence.idl
  - CFPortTypes_CharSequence.idl
  - CFPortTypes_DoubleSequence.idl
  - CFPortTypes_FloatSequence.idl
  - CFPortTypes_LongDoubleSequence.idl
  - CFPortTypes_LongLongSequence.idl
  - CFPortTypes_LongSequence.idl
  - CFPortTypes_ShortSequence.idl
  - CFPortTypes_UlongLongSequence.idl
  - CFPortTypes_UlongSequence.idl
  - CFPortTypes_UshortSequence.idl
  - CFPortTypes_WcharSequence.idl
  - CFPortTypes_WstringSequence.idl
- CF Standard Event CORBA Module IDL Files
  - CF_SE_DomainEvent.idl
  - CF_SE_StateEvent.idl

Software Radio PSM – DfSWRadio IDL Files

- DfSWRadio.idl
- Common Layer CORBA Module IDL Files
  - DfSWRadioCommonLayer.idl
  - DfSWRadioCommonLayerBasicTypes.idl
  - DfSWRadioErrorControl.idl
  - DfSWRadioErrorControlManagement.idl
  - DfSWRadioErrorControlSignal.idl
  - DfSWRadioFlowControl.idl
  - DfSWRadioFlowControlManagement.idl
  - DfSWRadioFlowControlSignaling.idl
  - DfSWRadioMeasurement.idl
  - DfSWRadioMeasurementManagement.idl
  - DfSWRadioMeasurementPoint.idl
  - DfSWRadioMeasurementRecorder.idl
  - DfSWRadioMeasurementStorage.idl
  - DfSWRadioMeasurementTypes.idl
  - DfSWRadioPDU.idl
  - DfSWRadioQoSManagement.idl
  - DfSWRadioStreamControl.idl

Software Radio PSM – DfSWRadio IDL Files

- Data Link Layer CORBA Module IDL Files
  - DfSWRadioDataLinkLayer.idl
  - DfSWRadioDataLinkLayerAckConnectionless.idl
  - DfSWRadioDataLinkLayerConnection.idl
  - DfSWRadioDataLinkLayerConnectionless.idl
  - DfSWRadioDataLinkLayerLocalManagement.idl
  - DfSWRadioDataLinkLayerMAC.idl
  - DfSWRadioDataLinkLayerTypes.idl
- Radio Set Management CORBA Module IDL Files
  - DfSWRadioControlRadioSetManagement.idl
- Physcial Layer CORBA Module IDL Files
  - DfSWRadioPhysicalLayer.idl

SW Radio PSM – XML Files

Component XML Descriptors

- DTDs
  - softwarecomponent.dtd
  - softpkg.dtd
  - properties.dtd
  - softwareassembly.dtd
  - devicepkg.dtd
  - deviceconfiguration.dtd
  - domainmanagerconfiguration.dtd
- SWRadio Schema
  - Properties.xsd
- CCM Schema
  - ComponentPackageDescription.xsd
  - ComponentInterfaceDescription.xsd
  - ComponentAssemblyDescription.xsd
- Communication Channel and Equipment Schema
  - CommChannel.xsd

Software Radio PSM – POSIX Profiles

- Two Profiles are defined which are a subset of the Real-time Controller System Profile (PSE52) Standardized Application Environment Profile - POSIX® Realtime Application Support (AEP), IEEE Std 1003.13-2003
  - The application environment profile (AEP), which is for constrained embedded general purpose processing, is the preferred profile for embedded processing and its utilization is encouraged for all processing environments.
  - The lightweight application environment profile (LwAEP) is more constrained than the AEP and is targeted towards environments with limited computing support such as embedded processors like Digital Signal Processors (DSPs) and micro-controllers.

PRISMTECH
Productivity Tools & Middleware

Software Radio Specification

SCA Relationship

Spectra Software Defined Radio Products

SCA Relationship

- The following slides will provide an overview of the OMG Software Radio spec and describe its relationship to the SCA in the following areas:
  - Specification Formation
  - Component Definition Flexibility
  - Specification Maturity
  - SCA Compatibility
  - Extensions

Specification Formation

- The SCA is expressed in platform specific technologies such as CORBA interfaces and XML DTDs.
- The OMG Software Radio specification is expressed in platform independent technologies using UML for PIM definitions and UML profiles for creating PIMs.
  - From its inception, the Software Radio specification was created to utilize UML 2.0 constructs and extension mechanisms.
  - The specification is based on OMG Model Driven Architecture principles, a key element of which is to express a language and middleware independent Platform Independent Model (PIM)
  - Sections are expressed in an industry consistent format: description, operations, attributes, constraints (Object Constraint Language (OCL)), types and exceptions, syntax, semantics
  - All model diagrams are illustrated using associations along with role descriptions.
  - PIM to PSM Transformation Rules stated so other PSMs can be defined.
  - Allows Waveform Designs to be captured independently of technology and transform into any technologies as technologies evolve.

Component Definition Flexibility

- Flexibility
  - Includes options for PIM level specification of Lightweight component definitions
    - Domain and Device Management allow for port definitions to specific interfaces which are supported
    - Lighter weight Device components support
      - *a configurable combination of states and statuses and*
      - *what capacities, if any, it manages*
    - Lighter weight Application Components than Resource Component can be defined. Minimal set of interfaces needed for deployment are: Lifecycle, PropertySet, PortSupplier, PortConnector
      - *If they exist for a component then deployment machinery uses them.*
  - IDL files are partitioned so only applicable interfaces are used for resource constraint systems

Specification Maturity

- Several years of development experience have provided insight into modeling and implementation optimizations beyond the SCA
  - Deployment Optimizations
    - Property configuration values are simplified
      - *Able to specify initial configuration values for all deployed components*
      - *At the Component level – property definitions only*
      - *At the Component Implementation level – denotes the actual property values for an implementation*
      - *At the Application Assembly level – component instantiation values to be used for initial configuration*

Specification Maturity

- Deployment Optimizations
  - Connection Behavior is simplified
    - Connections are managed at the component level not at uses port level
  - Able to retrieve a list of provided interfaces at one time.
  - Application Factory Component can make all devices' characteristic and capacities decisions
- Teardown Optimization
  - Disconnection Behavior – disconnections are only necessary for radio services (not deployed components)
- Clarification and Completeness
  - On Resource state behavior in relationship to Lifecycle and ControllableComponent interfaces
  - On Component, interface, and property Definitions
    - Associations with other components along with their roles

SCA Compatibility

- SCA Compatibility
  - Functionally Equivalent to SCA
    - Like the SCA defines Infrastructure and Application interfaces but these interfaces are represented as UML interfaces and can be transformed into any technology such as CORBA
  - XML DTDs are backwards compatible with corresponding SCA DTDs
    - Extensions added such as nested component, an implementation can be an assembly
    - More complex deployment requirements can be expressed.
    - Allows for any Service Component to be deployed and referenced in the Domain.
    - Better well form-ness rules on cardinality
  - Slight changes to Resource interface
    - Resource Component is model the same for it uses and provides ports but
      - *Resource now inherits the PortConnector interface and only provides out provided interfaces*
      - *Connection behavior can still be implemented at the uses port level inside the resource.*

SCA Compatibility

- SCA Compatibility
  - Like the SCA API supplement provides Component Facilities as a formal part of the specification
    - Common Layer
    - Component Definitions
      - *Serial*
      - *Audio*
    - Link layer
    - Physical Layer

SCA Compatibility

- SCA Compatibility
  - Slight changes to Device interface.
    - Optional Capacity Operations, only needed when device manages capacities
    - Optional StateManagement Interface
      - *The relationships, values, and state transitions amongst the adminState, operationalState, usageState, and attributes are described in the ISO/IEC International Standard 10164-2.*
      - *Added states attribute that return all states*
      - *optional SetAdminState operation to set admin state*
      - *Optional Admin State, level of support determine by adminStateRequestSupportedCharacterisitic*
    - Extensions to Execute operation
      - *Support for runtime requests*
      - *Support for user threads creation inside a non-user (OE) process*

SCA Compatibility

- SCA Compatibility
  - When all ports are realized, DeviceManager and DomainManager component implementations are equivalent to their SCA counterparts.
    - Provides Port can realize the same interfaces as the SCA DomainManger
  - POSIX Operating System Application Environment Profile (AEP)
    - RTF activity to reference 2003 POSIX specs.
  - When all of the individual CORBA CF and PortTypes IDL files are included they parallel the specification of the SCA CF and PortTypes IDL files.

Extensions

- Extends SCA Hardware support with additional System Engineering concepts
  - Communication Equipment Definition
  - Communication Channel
- Extends the SCA domain management concept with radio and channel management interfaces and components.
- Lightweight AEP for signal processing components

PRISMTECH
Productivity Tools & Middleware

Software Radio Specification

Conformance

Spectra Software Defined Radio Products

Conformance

- Simply put, conformance is defined at level of component and interface usage.
  - No requirement on what components are required for a radio set/system.
  - One needs to determine what radio components along with their interfaces are required for a software radio being built based upon the radio requirements and level of portability one is striving for.
- The OMG Software Radio specification defines three levels of portability like the SCA, which are at the:
  - Radio domain level,
  - Radio's node level and
  - Application level.

Radio Domain Level Conformance

- This conformance involves the Domain Manager/Radio Manager component and domain/radio management interfaces.
- This capability provides a benefit of managing a family of radios the same way within a company's radio product lines or across many different vendors' radios.
- This does not mean the Human Machine Interface/Human Computer Interface looks the same but the underlying implementations are using the domain management interfaces.

Radio Node Level Conformance

- This conformance involves the Device Manager, Logical Device, and Service Components along with these interfaces and descriptors.
- This level of portability is important when third party vendors are supplying hardware and the software to interface to their hardware to radio manufacturers or radio system integrators.

Application Level Conformance

- For an application this involves the Resource Component, Resource interfaces, properties, and descriptors.
- This level of portability is important when third party vendors are supplying waveform applications to radio manufacturers or radio system integrators.

Summary

- Provides a flexible Software Radio DSL and facilities to model and capture waveform and platform designs independently of technology that can be transformed to any technologies
- The Software Radio DSL can be easily extended or constrained for other domains
- The OMG Software Radio specification maps to the SCA