Date: September 2008

PIM and PSM for Smart Antenna Specification

FTF Beta 1

OMG Document Number: dtc/2008-09-05
Standard document URL: http://www.omg.org/spec/smartant/1.0/PDF
Associated Schema Files*: http://www.omg.org/spec/smartant/20080301
http://www.omg.org/spec/smartant/20080302

* original files: sbc/08-03-06 (XMI), sbc/08-03-09 (IDL)

This OMG document replaces the submission document (sbc/08-03-05, Alpha). It is an OMG Adopted Beta Specification and is currently in the finalization phase. Comments on the content of this document are welcome, and should be directed to *issues@omg.org* by August 29, 2008.

You may view the pending issues for this specification from the OMG revision issues web page *http://www.omg.org/issues/*.

The FTF Recommendation and Report for this specification will be published on November 10, 2008. If you are reading this after that date, please download the available specification from the OMG Specifications Catalog.

Copyright © 2007, HY-SDR Research center, Hanyang University
Copyright © 2007, L-3 Communications
Copyright © 2008, Object Management Group
Copyright © 2007, SDR Forum
Copyright © 2007, Virginia Tech

USE OF SPECIFICATION - TERMS, CONDITIONS & NOTICES

The material in this document details an Object Management Group specification in accordance with the terms, conditions and notices set forth below. This document does not represent a commitment to implement any portion of this specification in any company's products. The information contained in this document is subject to change without notice.

LICENSES

The companies listed above have granted to the Object Management Group, Inc. (OMG) a nonexclusive, royalty-free, paid up, worldwide license to copy and distribute this document and to modify this document and distribute copies of the modified version. Each of the copyright holders listed above has agreed that no person shall be deemed to have infringed the copyright in the included material of any such copyright holder by reason of having used the specification set forth herein or having conformed any computer software to the specification.

Subject to all of the terms and conditions below, the owners of the copyright in this specification hereby grant you a fully-paid up, non-exclusive, nontransferable, perpetual, worldwide license (without the right to sublicense), to use this specification to create and distribute software and special purpose specifications that are based upon this specification, and to use, copy, and distribute this specification as provided under the Copyright Act; provided that: (1) both the copyright notice identified above and this permission notice appear on any copies of this specification; (2) the use of the specifications is for informational purposes and will not be copied or posted on any network computer or broadcast in any media and will not be otherwise resold or transferred for commercial purposes; and (3) no modifications are made to this specification. This limited permission automatically terminates without notice if you breach any of these terms or conditions. Upon termination, you will destroy immediately any copies of the specifications in your possession or control.

PATENTS

The attention of adopters is directed to the possibility that compliance with or adoption of OMG specifications may require use of an invention covered by patent rights. OMG shall not be responsible for identifying patents for which a license may be required by any OMG specification, or for conducting legal inquiries into the legal validity or scope of those patents that are brought to its attention. OMG specifications are prospective and advisory only. Prospective users are responsible for protecting themselves against liability for infringement of patents.

GENERAL USE RESTRICTIONS

Any unauthorized use of this specification may violate copyright laws, trademark laws, and communications regulations and statutes. This document contains information which is protected by copyright. All Rights Reserved. No part of this work covered by copyright herein may be reproduced or used in any form or by any means--graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems--without permission of the copyright owner.

DISCLAIMER OF WARRANTY

WHILE THIS PUBLICATION IS BELIEVED TO BE ACCURATE, IT IS PROVIDED "AS IS" AND MAY CONTAIN ERRORS OR MISPRINTS. THE OBJECT MANAGEMENT GROUP AND THE COMPANIES LISTED ABOVE MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS PUBLICATION, INCLUDING BUT NOT LIMITED TO ANY WARRANTY OF TITLE OR OWNERSHIP, IMPLIED WARRANTY OF MERCHANTABILITY OR WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE OR USE. IN NO EVENT SHALL THE OBJECT MANAGEMENT GROUP OR ANY OF THE COMPANIES LISTED ABOVE BE LIABLE FOR ERRORS CONTAINED HEREIN OR FOR DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL, RELIANCE OR COVER DAMAGES, INCLUDING LOSS OF PROFITS, REVENUE, DATA OR USE, INCURRED BY ANY USER OR ANY THIRD PARTY IN CONNECTION WITH THE FURNISHING, PERFORMANCE, OR USE OF THIS MATERIAL, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

The entire risk as to the quality and performance of software developed using this specification is borne by you. This disclaimer of warranty constitutes an essential part of the license granted to you to use this specification.

RESTRICTED RIGHTS LEGEND

Use, duplication or disclosure by the U.S. Government is subject to the restrictions set forth in subparagraph (c) (1) (ii) of The Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or in subparagraph (c)(1) and (2) of the Commercial Computer Software - Restricted Rights clauses at 48 C.F.R. 52.227-19 or as specified in 48 C.F.R. 227-7202-2 of the DoD F.A.R. Supplement and its successors, or as specified in 48 C.F.R. 12.212 of the Federal Acquisition Regulations and its successors, as applicable. The specification copyright owners are as indicated above and may be contacted through the Object Management Group, 250 First Avenue, Needham, MA 02494, U.S.A.

TRADEMARKS

MDA®, Model Driven Architecture®, UML®, UML Cube logo®, OMG Logo®, CORBA® and XMI® are registered trademarks of the Object Management Group, Inc., and Object Management Group™, OMG™ , Unified Modeling Language™, Model Driven Architecture Logo™, Model Driven Architecture Diagram™, CORBA logos™, XMI Logo™, CWM™, CWM Logo™, IIOP™ , MOF™ , OMG Interface Definition Language (IDL)™ , and OMG Systems Modeling Language (OMG SysML)™ are trademarks of the Object Management Group. All other products or company names mentioned are used for identification purposes only, and may be trademarks of their respective owners.

COMPLIANCE

The copyright holders listed above acknowledge that the Object Management Group (acting itself or through its designees) is and shall at all times be the sole entity that may authorize developers, suppliers and sellers of computer software to use certification marks, trademarks or other special designations to indicate compliance with these materials.

Software developed under the terms of this license may claim compliance or conformance with this specification if and only if the software compliance is of a nature fully matching the applicable compliance points as stated in the specification. Software developed only partially matching the applicable compliance points may claim only that the software was based on this specification, but may not claim compliance or conformance with this specification. In the event that testing suites are implemented or approved by Object Management Group, Inc., software developed using this specification may claim compliance or conformance with the specification only if the software satisfactorily completes the testing suites.

OMG's Issue Reporting Procedure

All OMG specifications are subject to continuous review and improvement. As part of this process we encourage readers to report any ambiguities, inconsistencies, or inaccuracies they may find by completing the Issue Reporting Form listed on the main web page *http://www.omg.org*, under Documents, Report a Bug/Issue (http://www.omg.org/technology/agreement.htm).

Table of Contents

Preface

About the Object Management Group

OMG

Founded in 1989, the Object Management Group, Inc. (OMG) is an open membership, not-for-profit computer industry standards consortium that produces and maintains computer industry specifications for interoperable, portable and reusable enterprise applications in distributed, heterogeneous environments. Membership includes Information Technology vendors, end users, government agencies and academia.

OMG member companies write, adopt, and maintain its specifications following a mature, open process. OMG's specifications implement the Model Driven Architecture® (MDA®), maximizing ROI through a full-lifecycle approach to enterprise integration that covers multiple operating systems, programming languages, middleware and networking infrastructures, and software development environments. OMG's specifications include: UML® (Unified Modeling Language™); CORBA® (Common Object Request Broker Architecture); CWM™ (Common Warehouse Metamodel); and industry-specific standards for dozens of vertical markets.

More information on the OMG is available at *http://www.omg.org/*.

OMG Specifications

As noted, OMG specifications address middleware, modeling and vertical domain frameworks. A catalog of all OMG Specifications Catalog is available from the OMG website at:

http://www.omg.org/technology/documents/spec_catalog.htm

Specifications within the Catalog are organized by the following categories:

OMG Modeling Specifications

- UML
- MOF
- XMI
- CWM
- Profile specifications.

OMG Middleware Specifications

- CORBA/IIOP
- IDL/Language Mappings
- Specialized CORBA specifications
- CORBA Component Model (CCM).

Platform Specific Model and Interface Specifications

- CORBAservices
- CORBAfacilities
- OMG Domain specifications
- OMG Embedded Intelligence specifications
- OMG Security specifications.

All of OMG's formal specifications may be downloaded without charge from our website. (Products implementing OMG specifications are available from individual suppliers.) Copies of specifications, available in PostScript and PDF format, may be obtained from the Specifications Catalog cited above or by contacting the Object Management Group, Inc. at:

OMG Headquarters
140 Kendrick Street
Building A, Suite 300
Needham, MA 02494
USA
Tel: +1-781-444-0404
Fax: +1-781-444-0320
Email: *pubs@omg.org*

Certain OMG specifications are also available as ISO standards. Please consult *http://www.iso.org*

Typographical Conventions

The type styles shown below are used in this document to distinguish programming statements from ordinary English. However, these conventions are not used in tables or section headings where no distinction is necessary.

Times/Times New Roman - 10 pt.: Standard body text

Helvetica/Arial - 10 pt. Bold: OMG Interface Definition Language (OMG IDL) and syntax elements.

`Courier - 10 pt. Bold:` Programming language elements.

Helvetica/Arial - 10 pt: Exceptions

Note – Terms that appear in *italics* are defined in the glossary. Italic text also represents the name of a document, specification, or other publication.

Issues

The reader is encouraged to report any technical or editing issues/problems with this specification to *http://www.omg.org/technology/agreement.htm*.

1 Scope

This Specification respond to the requirements set by “Request for Proposals of PIM and PSM for Smart Antenna” (sbc/06-12-10) of smart antenna subsystem that can be utilized for expanding a single antenna system to array antenna system.

The Smart Antenna specification is physically partitioned into three major chapters: UML Profile for Smart Antenna (SA), SA PIM, and SA PSM. UML Profile for SA defines a language for modeling smart antenna system by expanding the UML language.

SA PIM provides a set of interfaces for interfacing with the signal processing module, RF module, and controller module. SA PSM provides a rule for transforming the elements of the profile and SA PIM into the platform specific model for CORBA IDL and XML

The SA specification is related to “Communication Channel and Equipment Specification (formal/07-03-02)” volume in such a way that stereotypes and class that have not been commented in the SA specification are defined in it.

2 Conformance

There are two kinds of conformance with respect to the SA profile: conformance on the part of a SA model and conformance on the part of a MDA tool.

2.1 Conformance by a Model of a Specific Application

A UML model of a specific SA either conforms to the SA profile or it does not. Such a UML model conforms to the SA profile if it satisfies all constraints imposed by the profile package.

2.2 Conformance by a Tool

2.2.1 Definition of Terms for Discussion of Tool Conformance

To support the discussion of conformance by a MDA tool, we define two terms: “identified subset of UML 2.0" and “all constructs defined by the profile.” The identified subset of UML 2.0 for the profile is the set of packages contained in the UML 2.0 Superstructure specification Part 1 (Structure). Part 1 includes the following packages and the transitive closure of all packages contained by these packages and of all packages upon which these packages depend:

- Classes
- Composite Structures
- Components
- Deployments

Here after we sometimes use the abbreviated term identified subset to refer to the identified subset of UML 2.0. The term all constructs defined by the profile is defined to mean all constructs that are part of the package’s identified subset of UML 2.0, plus all extensions to that subset that the profile defines. Thus this term includes UML constructs that are part of the identified subset but that are not extended by the profile.

2.2.2 Categories of Tool Conformance

A tool is considered to be a conformant simple modeling tool for the communication channel profile if it does both of the following:

- Supports expression of all constructs defined by the profile, via UML 2.0 notation.
- Supports the UML 2.0 XMI exchange mechanism for the identified subset and for UML 2.0 profiles.

A tool is considered to be a conformant CORBA/XML-based forward engineering tool for the profile if it does the following:

- Supports the PIM-to-PSM Mapping defined in Chapter 9.
- Produces comm channel manager components PSMs that are conformant to the behavior defined in the PIM.

Alternately, if a tool only produces a component skeleton, the skeleton must not make it impossible for a full component based on the skeleton to qualify as a conformant component – in other words, the skeleton must be able to form the basis of a conformant component.

A forward engineering tool that targets a platform technology other than CORBA/XML can legitimately claim a degree of conformance to the communication channel profile and PIM derived from the Profile if it conforms to the PIM-to-PSM Mapping and produces components PSMs that are conformant components to the behavior in defined in the PIM, or produces component skeletons that can form the basis of conformant components. In practice this requires the definition of an alternate PIM-PSM mapping.

A forward engineering tool of this nature for the platform “X” is considered to be a conformant X-Based forward engineering tool for the profile.

2.3 Conformance on the part of a Component PSM

The interfaces and components as defined in sections 7 and 8 of this specification are not required to be used for a given platform or application. A platform or application uses the interfaces and component definitions that meet their needs. Conformance is at the level of usage as follows:

- A PSM implementation (no matter what language) of an interface defined in this specification needs to be conformant to the interface definition as described in the specification.
- A PSM implementation (no matter what language) of a component defined in this specification needs to be conformant to the component definition (ports, interfaces realized, properties, etc.) as described in the specification.

A component is considered to be a conformant for CORBA/XML platform if it does all of the following:

- Implements the CORBA interfaces that the component PSM defines
- Implements the XML serialization formats that the component PSM defines.
- Implements the semantics that the component PIM defines.

Note that the component PIM essentially defines the semantics for the CORBA interfaces and XML serialization formats. The semantics for a CORBA interface defined in the component PSM are defined by the semantics of the corresponding element(s) in the component PIM. It is possible to deduce the corresponding elements in the PIM for such a CORBA interface by reversing the PIM-PSM Mapping.

3 References

3.1 Normative References

3.1.1 UML and Profile Specifications

3.1.1.1 UML Language Specification

Unified Modeling Language (UML) Superstructure Specification Version 2.1.2
Formal OMG Specification, document number: formal/2007-11-02
The Object Management Group, November 2007
[http://www.omg.org]

Unified Modeling Language (UML) Infrastructure Specification Version 2.1.2
Formal OMG Specification, document number: formal/2007-11-04
The Object Management Group, November 2007
[http://www.omg.org]

3.1.1.2 OCL Language Specification

Object Constraint Language (OCL) Specification Version 2.0
Formal OMG Specification, document number: formal/2006-05-01
The Object Management Group, May 2006
[http://www.omg.org]

3.1.1.3 UML Profile for CORBA Specification

UML Profile for CORBA Specification Version 1.0
Formal OMG Specification, document number: formal/2002-04-01
The Object Management Group, April 2002
[http://www.omg.org]

3.1.1.4 MOF 2.0/XMI Mapping Specification

Meta Object Facility (MOF) 2.0 XMI Mapping Specification, Version 2.1.1
Formal OMG Specification, document number: formal/2007-12-01
The Object Management Group, December 2007
[http://www.omg.org]

3.1.2 CORBA Core Specifications

3.1.2.1 CORBA Specification

Common Object Request Broker (CORBA/IIOP), Version 3.1
Formal OMG Specification, document number: formal/2007-12-01
The Object Management Group, December 2007
[http://www.omg.org]

3.1.2.2 Real-time CORBA Specifications

3.1.2.2.1 Real-time CORBA Specifications (Dynamic Scheduling)

Real-time - CORBA Specification (Dynamic Scheduling), Version 2.0
Formal OMG Specification, document number: formal/2003-11-01
The Object Management Group, November 2003
[http://www.omg.org]

3.1.2.2.2 Real-time CORBA Specifications (Static Scheduling)

Real-time - CORBA Specification (Static Scheduling), Version 1.2
Formal OMG Specification, document number: formal/2005-01-04
The Object Management Group, January 2005
[http://www.omg.org]

3.1.2.3 CORBA/e Specification

CORBA/e Specification
Draft Adopted OMG Specification, document number: ptc/06-08-03
The Object Management Group, August 2006
[http://www.omg.org]

3.1.3 UML Models

3.1.3.1 UML Profile for Communication Channel

UML Profile for Communication Channel XMI File
Formal OMG document number: formal/07-03-07
The Object Management Group, March 2007
[http://www.omg.org]

3.1.3.2 UML Profile for Component Framework

UML Profile for Component Framework XMI File
Formal OMG document number: formal/07-03-07
The Object Management Group, March 2007
[http://www.omg.org]

3.1.3.3 Common and Data Link Layer Facilities PIM

Common and Data Link Layer Facilities PIM XMI File
Formal OMG document number: formal/07-03-07
The Object Management Group, March 2007
[http://www.omg.org]

3.2 Non-normative References

3.2.1 Common and Data Link Layer Facilities Specification, v1.0

Common and Data Link Layer Facilities Specification
Formal OMG document number: formal/2007-03-05
The Object Management Group, March 2007
[http://www.omg.org]

3.2.2 UML Profile for Component Framework Specification, v1.0

Component Framework Specification
Formal OMG document number: formal/2007-03-05
The Object Management Group, March 2007
[http://www.omg.org]

3.2.3 Communication Channel and Equipment Specification, v1.0

Communication Channel and Equipment Specification
Formal OMG document number: formal/2007-03-02
The Object Management Group, March 2007
[http://www.omg.org]

4 Terms and Definitions

For the purposes of this specification, the terms and definitions given in the normative reference and the following apply.

For the purposes of this document, the following terms and definitions apply.

Common Object Request Broker Architecture (CORBA)

An OMG distributed computing platform specification that is independent of implementation languages.

Component

A component can always be considered an autonomous unit within a system or subsystem. It has one or more ports, and its internals are hidden and inaccessible other than as provided by its interfaces. A component represents a modular part of a system that encapsulates its contents and whose manifestation is replaceable within its environment. A component exposes a set of ports that define the component specification in terms of provided and required interfaces. As such, a component serves as a type, whose conformance is defined by these provided and required interfaces (encompassing both their static as well as dynamic semantics).

Facility

The realization of certain functionality through a set of well defined interfaces.

Interface Definition Language (IDL)

An OMG and ISO standard language for specifying interfaces and associated data structures.

Metadata

The Data that represents a models. For example, a UML model; a CORBA object model expressed in IDL; and a relation database schema expressed using CWM.

Metamodel

A model of models

Model

A formal specification of the function, structure and/or behavior of an application or system.

Model Driven Architecture (MDA)

An approach to IT system specification that separates the specification of functionality from the specification of the implementation of that functionality on a specific technology platform.

Platform

A set of subsystems/technologies that provide a coherent set of functionality through interfaces and specified usage patterns that any subsystem that depends on the platform can use without concern for the details of how the functionality provided by the platform is implemented.

Platform Independent Model (PIM)

A model of a subsystem that contains no information specific to the platform, or the technology that is used to realize it.

Platform Specific Model (PSM)

A model of a subsystem that includes information about the specific technology that is used in the realization of it on a specific platform, and hence possibly contains elements that are specific to the platform.

Request for Proposal (RFP)

A document requesting OMG members to submit proposals to the OMG's Technology Committee. Such proposals must be received by a certain deadline and are evaluated by the issuing task force.

Unified Modeling Language (UML)

An OMG standard language for specifying the structure and behavior of systems. The standard defines an abstract syntax and a graphical concrete syntax.

UML Profile

A standardized set of extensions and constraints that tailors UML to particular use.

5 Symbols and abbreviated terms

Abbreviation	Definition
CORBA	Common Object Request Broker Architecture
DOA	Direction Of Arrival
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
GPP	General Purpose Processor
I/O	Input/Output
IDL	Interface Definition Language
IIOP	Internet Inter-ORB Protocol
ISO	International Standards Organization
N/A	Not Applicable
OMG	Object Management Group
ORB	Object Request Broker
OS	Operating System
PIM	Platform Independent Model
PSM	Platform Specific Model
RF	Radio Frequency
SA	Smart Antenna
SDR	Software Defined Radio
SWRadio	Software Radio Components
UML	Unified Modeling Language
XML	eXtensible Markup Language

6 Additional Information

6.1 Changes to Adopted OMG Specifications

The specifications contained in this document require no changes to adopted OMG specifications.

6.2 Guide to this Specification

This specification consists of three major parts, contained in the following chapters 7 to 9.

- Chapter 7 defines the modeling language used in this specification in form of a UML profile for smart antenna components.

- Chapter 8 contains the Smart Antenna Facilities Platform Independent Model (PIM). The UML language defined in Chapter 7 is used to specify this PIM.
- In Chapter 9, the mapping process from the Platform Independent Model (PIM) to a Platform Specific Model (PSM) is described.

6.3 Acknowledgements

The following organizations (listed in alphabetical order) contributed to this specification:

- BAE Systems
- Hanyang University
- L-3 Communications
- MITRE
- PrismTech
- Raytheon
- SDR Forum
- Virginia Tech University

6.4 Security and Regulatory

There are two subjects that are not included herein: Security and Regulatory. These subjects impact the broader software radio (and other device-bound) topics (sensors, robotics, etc.). As such, this document will, in a future release, address a specialization of the general solution adopted by the broader security and regulatory standardization community. Therefore the following sections apply:

6.4.1 Security

This architecture document does not include the functionality for security regarding certification of the source code or over the air delivery of new characteristics for the Smart Antenna or security functional protection mechanisms. The addition of such mechanisms is not expected to alter the architecture defined herein.

6.4.2 Regulatory

This architecture document does not include the functionality for managing regulatory requirements for the Smart Antenna. The Smart Antenna herein described functionality could most likely require the addition of finer control and explicit regulatory controls. The addition of such controls is not expected to alter the architecture defined herein; although finer power/bandwidth/ERP may be required to meet regulations.

6.5 Smart Antenna System

A Smart antenna is an antenna array system that is aided by a processing system that processes the signals received by the array or transmitted by the array using suitable array algorithms to improve wireless system performance. An antenna array consists of a set of distributed antenna elements (dipoles, monopoles or directional antenna elements) arranged in certain geometry (e.g., linear, circular, or rectangular grid) where the spacing between the elements can vary. The signals collected by individual elements are coherently combined in a manner that increases the desired signal strength and reduces the interference from other signals. A smart antenna can be viewed as a combination of antenna elements, using some form of RF, IF or baseband array combination, that transmit or receive RF signals using “smart” algorithms. A software defined smart antenna is a smart antenna in which certain operating characteristics, such as the field of regard, frequency of operation, access mode, or transmit/receive waveforms can be altered by firmware or software download after its manufacture.

6.6 Classification of Smart Antennas

Base on the signal processing technique followed at the baseband output of the antenna array smart antennas can be group into four basic types based on : 1) Beamforming 2) Space time equalization 3) Diversity combining 4) Multiple input multiple output(MIMO) processing.

6.6.1 Beamforming

Through Beamforming, a smart antenna algorithm can receive predominantly from a desired direction (direction of the desired source) compared to some undesired direction(direction of interfering sources). This implies that the digital processing has the ability to shape the radiation pattern for both reception and transmission and to adaptively steer beams in the direction of the desired signals and put nulls in the direction of the interfering signals. This enables low co-channel interference and large antenna gain to the desired signal. Beamforming systems can be implemented in two ways; fixed beamforming systems or fully adaptive systems. A fixed beamforming system has a beamforming network(BFN) followed by RF switches which operate in the RF/analog domain. The switches are controlled by a control logic which selects a particular beam. Here the processing required is minimal as the control logic has to choose one of the predetermined set of weights to select a beam. In adaptive beamforming, the antenna gains or weights are chosen adaptively through running array algorithms in the digital domain.

6.6.2 Space-Time Equalization

The preceding two techniques usually assume that the signal of interest is a narrowband signal compared to the coherence bandwidth of the channel and is thus subjected to flat fading across the bandwidth of the signal. Multipath fading in wireless communication can also introduce frequency distortion to the received signal. By introducing temporal processing in each antenna element to remove the effect of frequency distortion and doing a spatial combining described above results in mitigating channel induced frequency selective fading and providing antenna gain. Such schemes are called space-time adaptive processing (STAP) or equalization.

6.6.3 Diversity Combining

A major limiting factor in wireless communication is multipath fading where the amplitude of the received signal fluctuates over time. The occurrence of a deep fade where the signal amplitude becomes very small can impair the communications link for a conventional or a single antenna system. When multiple antennas are used it becomes less likely that two or more antennas undergo deep fades at the same time. This diversity in the received signal, for the same transmitted information, is exploited by smart antenna processing schemes. Many simple algorithms, such as maximal ratio combining, equal gain combining, and selection diversity have been developed to take advantage of using antenna arrays to exploit diversity reception in wireless systems. These algorithms weight the received signal similar to beamforming but based on a different criterion used in the algorithm

6.6.4 Multiple Input Multiple Output (MIMO)

As the name suggests this scheme requires array processing at the transmitter and receiver. There are two different types of MIMO schemes: one uses spatial multiplexing to enhance data rate for a given bandwidth (thus, the spectral efficiency) and the other uses space time coding using diversity combining techniques to combat fading. In the multiplexing scheme, data is serial to parallel converted and transmitted simultaneously over multiple antenna elements. The receiver also uses multiple antenna elements to receive the signal and applies a maximum likelihood (ML) algorithm to retrieve the simultaneously transmitted symbols. One key assumption in this case is that the propagation environment has to provide rich scattering; in other words, the propagation channel has to include a large number of scattering objects that will generate independent fading at the antenna elements. In the case of space-time coding, symbols to be transmitted are coded over multiple antennas and symbol time durations in such a way that the receiver can easily regenerate the transmitted signals by doing a linear processing on received signal. The space-time codes rely on the orthogonality present in the coded symbols for proper detection, and additionally they require the fading to be independent between the antenna elements for best performance results.

7 UML Profile for Smart Antenna

This section defines the UML Profile for only Smart Antenna. The set of stereotypes and types that are not described in this section are defined in UML Profile for SWRadio components.

7.1 Types

- `Complex (real: Float, imag: Float)`
  Complex data type denote a general complex number.

- `ComplexSequence`
  ComplexSequence is an unbounded sequence of Complex (s)..

- `<<primitive>>ArrayAntennaType`
  ArrayAntennType,a specialization of String, denote the physical configuration of an array antenna(e.g., Phased Array, Circular Array, etc.)

7.2 CommEquipment for Smart Antenna

7.2.1 ArrayAntenna

Description

The ArrayAntenna stereotype, shown in Figure 7.1, represents an antenna array which consists of multiple antenna elements. The ArrayAntenna class shall have one or more AntennaElement's.

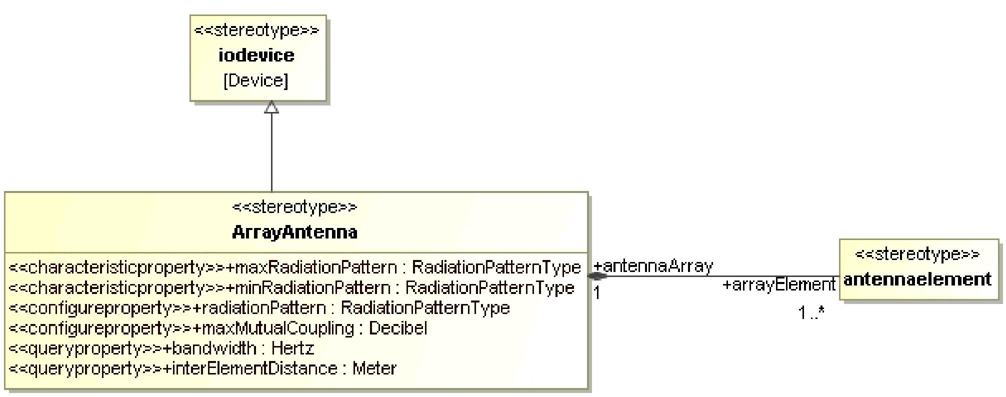

Figure 7.1 - ArrayAntenna Stereotype

Attributes

- `<<characteristicproperty>>maxRadiationPattern: RadiationPatternType`
  The maxRadiationPattern attribute indicates the maximum radiation pattern that the device is able to achieve.

- <<characteristicproperty>>minRadiationPattern: RadiationPatternType
  The minRadiationPattern attribute indicates the minimum radiation pattern that the device is able to achieve.
- <<configureproperty>>radiationPattern: RadiationPatternType
  The radiationPattern attribute represents the current radiation pattern configured in the device.
- <<characteristicproperty>>type: ArrayAntennaType
  The type attribute indicates the physical type of the array antenna
- <<configureproperty>>maxMutualCoupling: Decibel
  The maxMutualCoupling is maximum mutual coupling value between antenna elements.
- <<characteristicproperty>>bandwidth: Hertz
  The bandwidth attribute indicates the bandwidth of physical array antenna.
- <<characteristicproperty>>interElementDistance: Meter
  The interElementDistance attribute represents physical distance between antenna elements.
  The id attributes represents the identification of the channel.

M1 Associations

- arrayElemnt: AntennaElement [1..*]
  The individual radiating element object of the ArrayAntenna.

Constraint

An ArrayAntenna shall have at least one AnalogInputPort or one AnalogOutputPort.

8 Platform Independent Model (PIM)

The SA PIM provides interfaces used to configure and control a Smart Antenna Subsystem. In order to specify a SA PSM, we have to first define a standard PIM because a SA specification should be valid regardless of platform types. The SA Facilities has a dependency on the Communication Channel and Equipment Physical Layter Facilities as shown in Figure 8.2. The SA PIM consists of three facilities each of which has been defined in accordance with their functions. These three facilities operate in conjunction with each other to define the PIM for Smart Antenna.

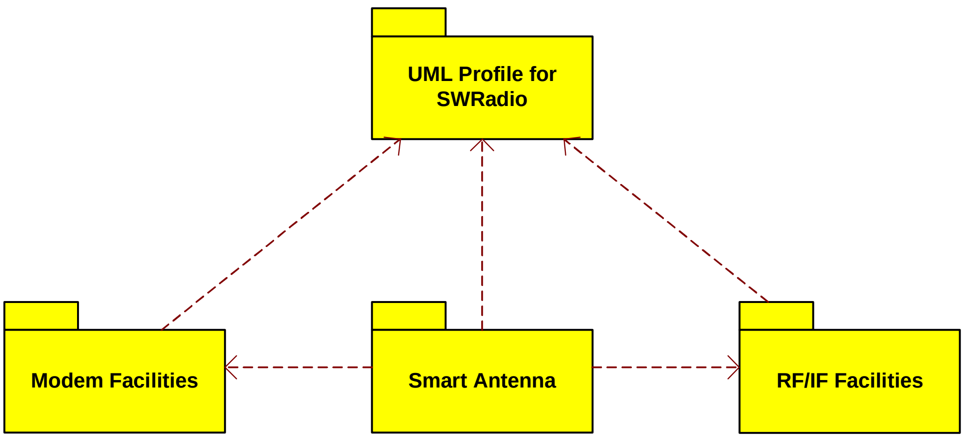

Figure 8.1 - Package Diagram

Three facilities are used to control the entire Smart Antenna Subsystem:

- Control Facilities
- Synchronization Facilities
- Algorithm Facilities

Figure 8.1 illustrates the relation of the three facilities of SA. The Control Facilities, which include SAControl component, RFControl interface, SynchronizationControl interface, and AlgorithmControl interface, are used to control all algorithm operations performed in the digital signal processing parts and RF/IF operations such as analog to digital or digital to analog conversion. The Synchronization Facilities, which include SASynchronization component, CalibrationComponent component, Calibration interface, SynchronizationComponent component, and Synchronization interface, are used for RF chain calibration and symbol (or frame) synchronization. CalibrationComponent component processes signals fed by RFIFComponent component and SynchronizationComponent component processes signals fed by ModemComponent component. The Algorithm Facilities, which include SAAlgorithm component and Algorithm components, i.e. BeamformingComponent, STCComponent, SpatialMultiplexingComponent, DOAEstimationComponent, and ChannelEstimationComponent, and interfaces, i.e., Beamforming, SpaceTimeCoding, SpatialMultiplexing, DOAEstimation, and ChannelEstimation, are used to execute all the algorithms that are needed for the Smart Antenna System to provide superb performance compared to Single Antenna System. Algorithm components process signals fed by ModemComponent component. The Smart Antenna Subsystem shall implement a single SAControl component and one or more Algorithm component(s). More detailed explanations about interfaces among facilities in the SA are provided in the following sections.

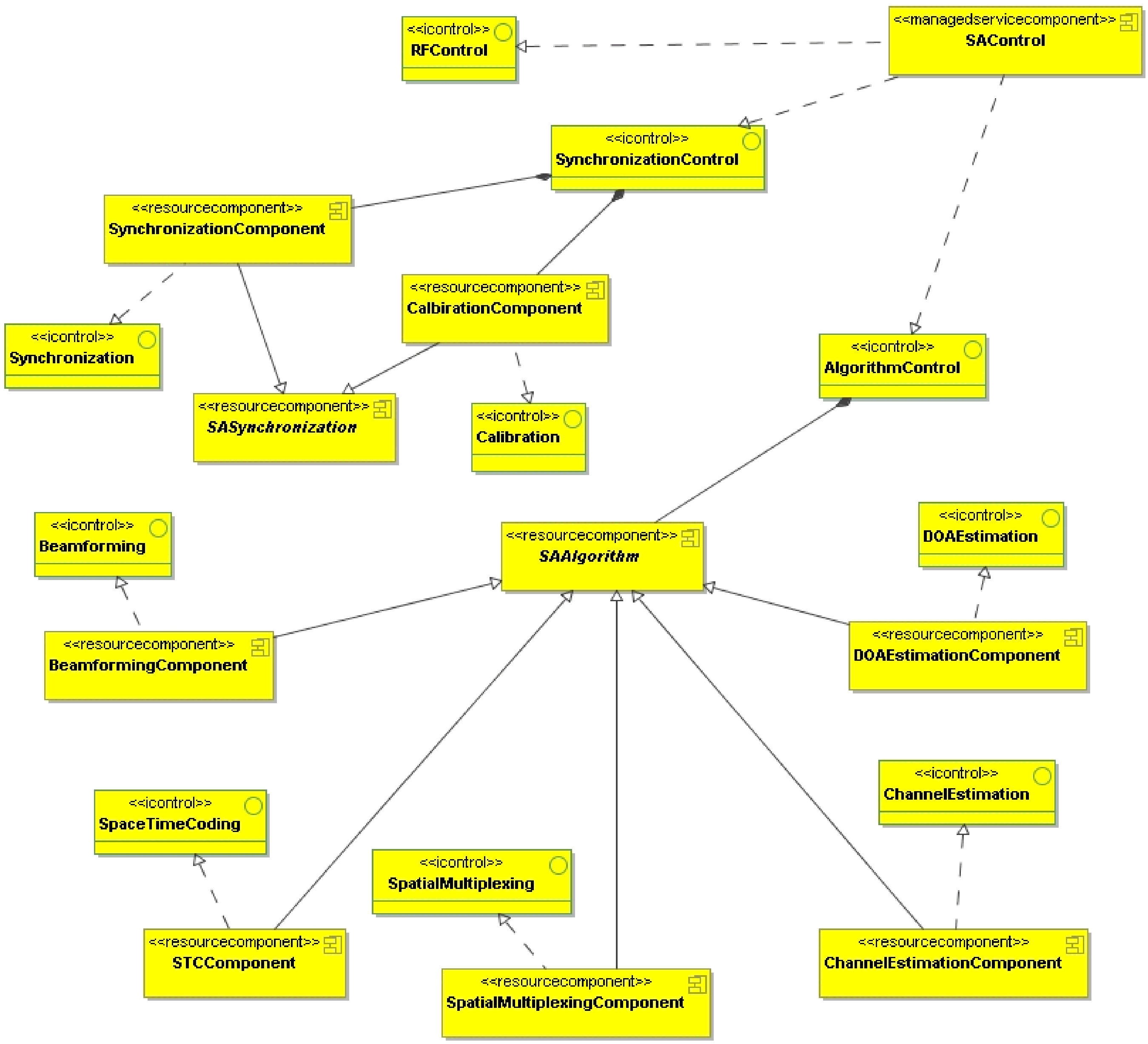

Figure 8.2 - SA Overview

8.1 Control Facilities

In this section, each function and interface provided by the Control Facilities is described. Figure 8.3 illustrates Control Facilities that include SAControl component, Synchronization Control interface, Algorithm Control interface, and RF Control interface. It can be observed from Figure 8.3 that RFControl interface, SynchronizationControl interface, and AlgorithmControl interface shall be realized by SAControl component, in order for SAControl component to control RFIFComponent component, SASynchronization component, and SAAlgorithm component, respectively, according to the functions to be performed in the SAControl component.

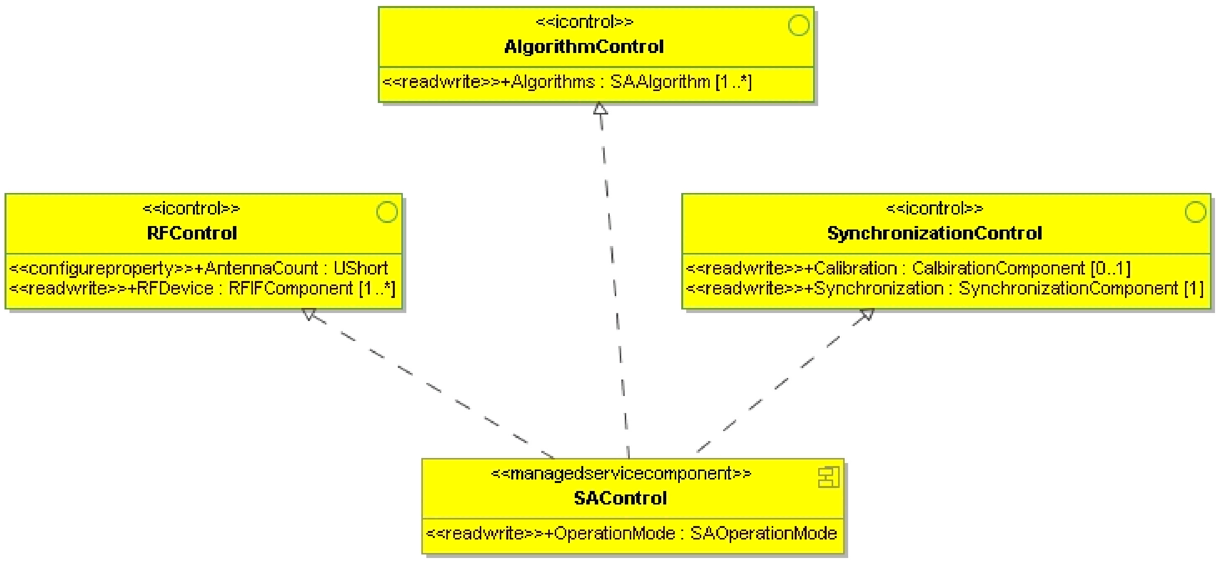

Figure 8.3 - Control Facilities

8.1.1 SAControl

Description

The SAControl component takes on the definition as described in the UML Profile for Component Framework:: Infrastructure::Service in addition to the realization of RFControl interface, AlgorithmControl interface, and SynchronizationControl interface. The SAControl component is used to control entire Smart Antenna Subsystem with state behavior.

Semantics

The SAControl's operational state shall be based upon the operational state of its Device components. The SAControl's usage state shall be IDLE when all of its Device components are IDLE. The SAControl's usage state becomes ACTIVE when any of its Device components is not IDLE. The SAControl's usage state shall be BUSY when all of its Device components are not IDLE. If the SAControl's administrative state is SHUTTING_DOWN or LOCKED, then its Device components shall be unavailable for application instantiation.

Attributes

- `<<readwrite>>OperationMode: SAOperationMode`
  The OperationMode attribute sets an operation mode of Smart Antenna Subsystem. The operation mode shall be one of three modes, TRANSMIT, RECEIVE, and COMBINATION. This attribute is only used when the SAControl's state is ACTIVE.

Types and Exceptions

`<<enumerationproperty>>SAOperationMode ( TRANSMIT, RECEIVE, COMBINATION )`
The SAOperationMode defines the operation mode of Smart Antenna Subsystem.
TRANSMIT: Smart Antenna Subsystem operates in transmitting mode.
RECEIVE: Smart Antenna Subsystem operates in receiving mode.
COMBINATION: Smart Antenna Subsystem operates in both transmitting and recieving mode.

8.1.2 AlgorithmControl

Description

The AlgorithmControl interface is used to control SAAlgorithm components.

Attributes

- `<<readwrite>>Algorithms: SAAlgorithm [1..*]`
  The AlgorithmControl interface shall contain the set of SAAlgorithm components. The SAAlgorithm references are used to control multiple SAAlgorithm components.

8.1.3 SynchronizationControl

Description

The SynchronizationControl interface is used to control the SynchronizationComponent and CalibrationComponent component.

Attributes

- `<<readwrite>>Synchronization: SynchronizationComponent [1]`
  The SynchronizationControl interface shall contain one SynchronizationComponent component. The SynchronizationComponent reference is used to control a SynchronizationComponent component.
- `<<readwrite>>Calibration: CalibrationComponent [0..1]`
  If the Smart Antenna Subsystem is implemented with beamforming, then the SynchronizationControl shall contain one CalibrationComponent, otherwise the CalibrationComponent may not be required. The CalibrationComponent reference is used to control a CalibrationComponent component.

8.1.4 RFControl

Description

The RFControl interface is used to control RFIFComponent components.

Attributes

- `<<readwrite>>RFDevices: RFIFComponent [1..*]`
  The RFControl interface shall contain the set of RFIFComponent components. The RFIFComponent references are used to control multiple RFIFComponent components.

- `<<configureproperty>>AntennaCount: UShort`
  The AntennaCount attribute represents the number of Antenna elements.

8.2 Synchronization Facilities

In this section each function and interface in Synchronization facilities is described. Figure 8.4 illustrates Synchronization Facilities that include the SASynchronization component, Calibration interface, CalibrationComponent component, SynchronizationComponent component, and Synchronization interface.

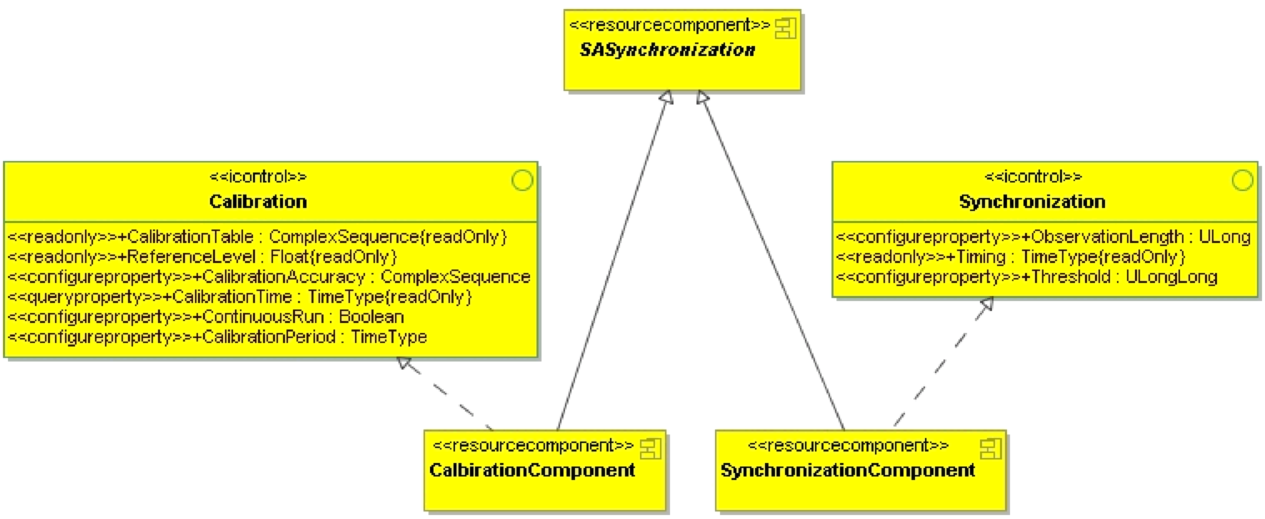

Figure 8.4 - Synchronization Facilities

8.2.1 SASynchronization

Description

The SASynchronization component is an abstract component from which the CalibrationComponent component and SynchronizationComponent component shall inherit.

Constraint

The SASynchronization component shall provide one ControlPort and at least one DataControlPort or DataPort.

8.2.2 Calibration

Description

The Calibration interface is used to calibrate whole RF/IF chains of the Smart Antenna System.

Attributes

- `<<readonly>>CalibrationTable: ComplexSequence`
  The CalibrationTable attribute represents the output of Calibration.

- `<<readonly>>ReferenceLevel: Float`
  The ReferenceLevel attribute represents the value normalized to a unit power or some non-unit input level.

- `<<configureproperty>>ContinuousRun: Boolean`
  The ContinuousRun attribute indicates whether or not calibration is executed continuously.

- `<<configureproperty>>CalibrationAccuracy: ComplexSequence`
  The CalibrationAccuracy attribute represents the required accuracy of calibration. The required accuracy shall be configured in both amplitude and phase.

- `<<configureproperty>>CalibrationPeriod: TimeType`
  The CalibrationPeriod attribute is used to control calibration period.

- `<<queryproperty>>CalibrationTime: TimeType`
  The CalibrationTime attribute return the time required for a single calibration processing using the active settings.

8.2.3 CalibrationComponent

Description

The CalibrationComponent component realizes Calibration interface and extends SASynchronization component. Calibration is to compensate for amplitude and phase differences of the RF/IF chain associated with each antenna in transmit and receive mode. Problem of calibration has arisen because the amplitude and phase characteristics of signal path associated with each antenna are different from each other. Especially even if the optimal weight vector is computed from the received signal such that the uplink communication of the smart antenna system can fully exploit the enhancements in both communication capacity and cell coverage, downlink beam-forming can never be optimized without accurate calibration. In other words, the objective of calibration is to compensate for the mutual coupling effects between antenna array elements as well as for the mismatches of channel amplitude and/or channel phase in Smart Antenna Systems.

8.2.4 Synchronization

Description

The Synchronization interface is used for symbol (or frame) synchronization of the Smart Antenna Subsystem.

Attributes

- `<<configureproperty>>ObservationLength: ULong`
  The ObservationLength attribute is used to configure observation length in samples.

- `<<readonly>>Timing: TimeType`
  The Timing attribute represents acquired symbol (or frame) timing.

- `<<configureproperty>>Threshold: ULongLong`
  The Threshold attribute is used to configure Threshold for signal detection. This value shall be normalized to an input level without dimension.

8.2.5 SynchronizationComponent

Description

The SynchronizationComponent component realizes Synchronization interface and extends SASynchronization component. Symbol (or frame) synchronization is a processing for detection of symbol (or frame) timing. Synchronization is performed prior to demodulation of symbol (or decoding of frame) and operation of smart antenna algorithm. To enhance the performance of Smart Antenna System, accurate symbol (or frame) timing shall be provided. In addition, to guarantee the QoS (Quality of Service) of initial network access, fast and robust acquisition of initial access signal shall be provided to the Smart Antenna System.

8.3 Algorithm Facilities

In this section each function and interface in Algorithm facilities is described. Figure 8.5 illustrates Algorithm Facilities that include the SAAlgorithm component, Algorithm components, which are BeamformingComponent, STCComponent, SpatialMultiplexingComponent, DOAEstimationComponent, and ChannelEstimationComponent, and interfaces, which are Beamforming interface, SpaceTimeCoding interface, SpatialMultiplexing interface, ChannelEstimation interface, and DOAEstimation interface.

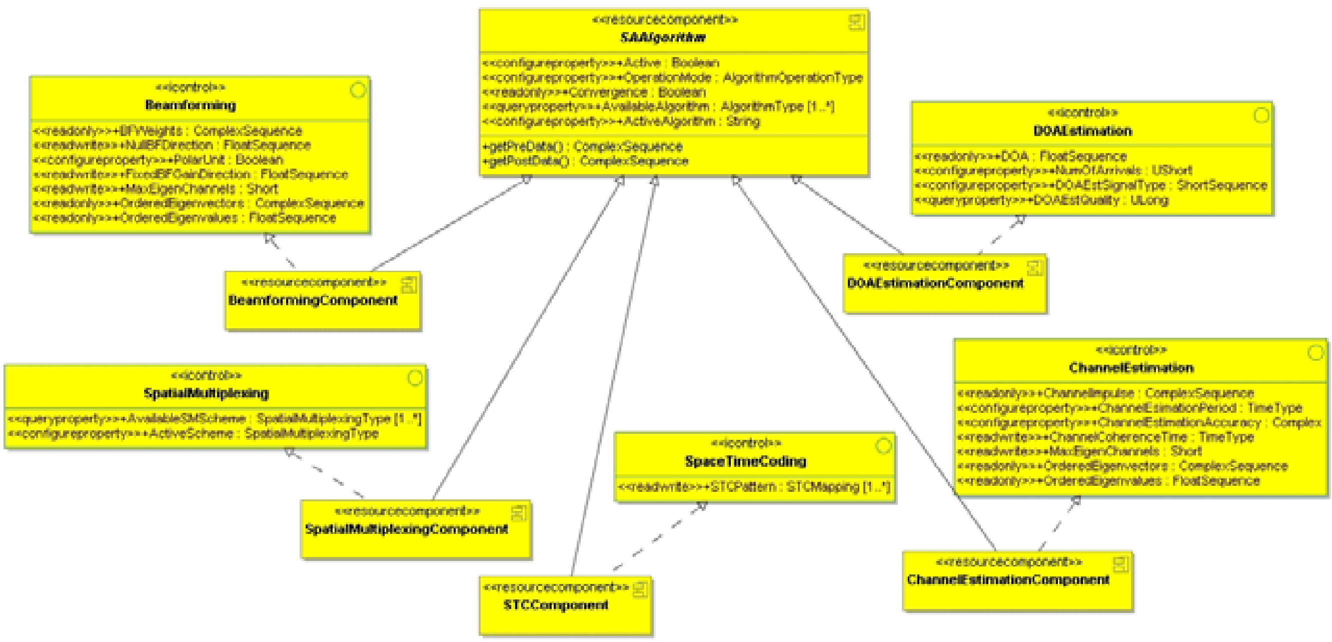

Figure 8.5 - Algorithm Facilities

8.3.1 SAAlgorithm

Description

The SAAlgorithm component is an abstract component from which all components in the Algorithm Facilities shall inherit. In other words, this component provides every interface for controlling all the Algorithm components.

Attributes

- <<configureproperty>>Active: Boolean
  The Active attribute indicates that the SAAlgorithm component is activated.

- <<configureproperty>>OperationType: AlgorithmOperationType
  The OperationType attribute sets an operation type of SAAlgorithm components. The operation type shall be one of three types, CONTUINUOUS, SINGLE_BURST, and REPEATED_BURST.

- <<readonly>>Convergence: Boolean
  The Convergence attribute indicates whether the algorithm is confident that its link quality is high enough to satisfy the configured QoS.

- <<queryproperty>>AvailableAlgorithm: AlgorithmType[1..*]
  The AvailableAlgorithm attribute represents a list of the algorithms available for use on SAalgorithm.

- <<configureproperty>>ActiveAlgorithm: String
  The ActiveAlgorithm attribute sets algorithm or gets activated algorithm.

Operation

- getPreData(return ComplexType)
  The getPreData operation is provided to command the SAAlgorithmDevice component to get pre-processing data such as pre-despreading data and pre-FFT data, etc.

- getPostData(return ComplexType)
  The getPreData operation is provided to command the SAAlgorithmDevice component to get post-processing data such as post-despreading data and post-FFT data, etc.

Types and Exceptions

- <<enumerationproperty>>AlgorithmOperationType (CONTINUOUS, SINGLE_BURST, REPEATED_BURST)
  The AlgorithmOperationType defines the operation type of data processing.
  CONTINUOUS: SAAlgorithm components process input signals continuously.
  SINGLE_BURST: SAAlgorithm components process the single burst input signal.
  REPEATED_BURST: SAAlgorithm components process repetitively the single burst input signal.

- AlgorithmType(Name: String, Delay: TimeType, PowerConsumption: Float, TolerableBandwidth: Hertz)
  Name: The name of an algorithm as a String.
  Delay: The time for an algorithm to perform a single execution.
  PowerConsumption: The power consumption for an algorithm to perform a single excution.
  TolerableBandwidth: The tolerable bandwidth for an algorithm to converge.

Constraint

The SAAlgorithm component shall provide one ControlPort and at least one DataControlPort or DataPort.

8.3.2 Beamforming

Description

The Beamforming interface is used to control the BeamformingComponent.

Attributes

- `<<readonly>>BFWeights: ComplexSequence`
  The BFWeights attribute is weight vectors computed by the BeamformingComponent. When this attribute is read, the BeamformingComponent computes a new value from received signals. When this attribute is set up, the BeamformingComponent applies a given value that provides the desired radiation pattern.

- `<<configureproperty>>PolarUnit: Boolean`
  The PolarUnit attribute is used to switch between real/imag mode and mag/phase mode of Weights attribute.

- `<<readwrite>>NullBFDirection: FloatSequence`
  The NullBFDirection attribute is used to specify directions of nulls in degree to block known source of interference.

- `<<readwrite>>FixedBFGainDirection: FloatSequence`
  The FixedBFGainDirection attribute is used to specify fixed gains (dB) in fixed direction (degree) to amplify weak signals in known direction.

- `<<readwrite>>SideLobeLevel: FloatSequence`
  The SideLobeLevel attribute is used to limit the side lobe level in decibel (dB).

- `<<readonly>>OrderedEigenvalues: FloatSequence`
  The OrderedEigenvalues attribute presents the eigen channel quality metrics used in BeamformingComponent.

- `<<readonly>>OrderedEigenvectors: ComplexSequence`
  The OrderedEigenvectors attribute presents the eigen channel quality metrics used in BeamformingComponent.

- `<<readwrite>>MaxEigenChannels: Short`
  The MaxEigenChannels attribute configures maximum number of eigen channels to be used in BeamformingComponent.

8.3.3 BeamformingComponent

Description

The BeamformingComponent component extends the SAAlgorithm component and realizes the Beamforming interface. A beamforming algorithm in the BeamformingComponent computes the weight vectors for both RX and TX operations. The weight vectors adaptively steer beams along the direction of desired signals and puts nulls along the direction of interfering signals..

8.3.4 SpaceTimeCoding

Description

The SpaceTimeCoding interface is used to control STCComponent.

Attributes

- `<<readwrite>>STCPattern: STCMapping[1..*]`
  The STCPattern attribute represents the actual definition of the STC mapping. Each input symbol of STCComponent is mapped to one of transmit antennas according to STCMapping.

Types and Exceptions

- `STCMapping ( NumAnt: UShort, codePattern: UShort [1..*])`
  NumAnt: Number of transmit antenna.
  codePattern: The actual space time code pattern as a UShort.

8.3.5 STCComponent

Description

The STCComponent component extends the SAAlgorithm component and realizes the SpaceTimeCoding interface. The STCComponent is for Space Time Coding (STC) processing. A Space Time Coding (STC) is a method employed to improve the reliability of data transmission in wireless communication systems using multiple transmit antennas. STCs rely on transmitting multiple, redundant copies of a data stream to the receiver in the hope that at least some of them may survive the physical path between transmission and reception in a good enough state to allow reliable decoding.

8.3.6 SpatialMultiplexing

Description

The SpatialMutiplexing interface is used to control SpatialMultiplexingComponent.

Attributes

- `<<queryproperty>>AvailableSMScheme: SpatialMultiplexingType[1..*]`
  The AvailableSMScheme attribute represents a list of the spatial multiplexing schemes available for use on SpatialMultiplexingComponent.

- `<<configureproperty>>ActiveSMScheme: SpatialMultiplexingType`
  The ActiveSMScheme attribute sets SpatialMultiplexingType or gets activated SpatialMultiplexingType.

Types and Exceptions

- `<<Primitive>>SpatialMultiplexingType`
  The SpatialMultiplexingType, a specialization of String, denotes the type of algorithm used for spatial multiplexing.(e.g.,V-BLAST, D-BLAST, H-BLAST, etc.).

8.3.7 SpatialMultiplexingComponent

Description

The SpatialMultiplexingComponent component extends the SAAlgorithm component and realizes the SpatialMultiplexing interface. SpatialMultiplexingComponent is for spatial multiplexing. The spatial multiplexing is a transmission technique in MIMO wireless communication to transmit independent and separately encoded data signals from each of the multiple transmit antennas.

8.3.8 ChannelEstimation

Description

The ChannelEstimation interface is used to control the ChannelEstimationComponent.

Attributes

- `<<readonly>>ChannelImpulse: ComplexSequence`
  The ChannelImpulse attribute represents the channel estimation vector that is calculated by the ChannelEstimationComponent component.
- `<<readwrite>>ChannelCoherenceTime: TimeType`
  The ChannelCoherenceTime attribute represents channel coherence time.
- `<<configureproperty>>ChannelEstimationPeriod: TimeType`
  The ChannelEstimationPeriod attribute is used to control channel estimation period. This attribute would be especially important to trade overhead time and processing against the rate of change in the channel due to platform motion, etc.
- `<<configureproperty>> ChannelEstimationAccuracy: Complex`
  The ChannelEstimationAccuracy attribute represents the required accuracy of channel estimation.
- `<<readonly>>OrderedEigenvalues: FloatSequence`
  The OrderedEigenvalues attribute presents the eigen channel quality metrics used in ChannelEstimationComponent.
- `<<readonly>>OrderedEigenvectors: ComplexSequence`
  The OrderedEigenvectors attribute presents the eigen channel quality metrics used in ChannelEstimationComponent.
- `<<readwrite>>MaxEigenChannels: Short`
  The MaxEigenChannels attribute configures maximum number of eigen channels to be used in ChannelEstimationComponent.

8.3.9 ChannelEstimationComponent

Description

The ChannelEstimationComponent component extends the SAAlgorithm component and realizes the ChannelEstimation interface. Space-time equalization system or diversity combining system is implemented using the ChannelEstimationComponent. The Space-time Equalization is a receiving technique which makes use of temporal processing on the signals received from multiple antennas to correct frequency distortion in the received signal path. And the diversity combining is another receiving technique to mitigate the multipath fading effects which are inherent in practical wireless networks by combining the signals of multiple antennas.

8.3.10 DOAEstimation

Description

The DOAEstimation interface is used to control the DOAEstimationComponent.

Attributes

- `<<readonly>>DOA: FloatSequence`
  The DOA attribute represents direction of arrival (DOA) angle in degree.
- `<<configureproperty>>NumOfArrivals: UShort`
  The NumOfArrivals specifies how many DOA estimates are required allowing for estimation of the arrival of the same signal from multiple directions.
- `<<configureproperty>>DOAEstSignalType: ShortSequence`
  The DOAEstimationSignalType attribute specifies the character of the various signals to estimate.
- `<<queryproperty>>DOAEstQuality: ULong`
  The DOAEstQuality attribute indicates DOA estimation quality.

8.3.11 DOAEstimationComponent

Description

The DOAEstimationComponent component extends the SAAlgorithm component and realizes the DOAEstimation interface.

9 PSM for Smart Antenna

9.1 Mapping Rule

This section defines a reference PSM that consists of the normative CORBA interface and the normative XML that are based upon the PIM and UML Profile for Smart Antenna. The PIM to PSM transformation rules are not universal rules for creating *any* PSM, but only used for the purpose of this specification. Non-CORBA PSMs may also be fully compliant to this specification as a whole. The rule set for transforming Smart Antenna PIM (UML packages, interfaces, types, and exceptions) to the CORBA interface and the XML is as follows:

1. UML interfaces and interface extensions are mapped to CORBA interfaces. The CORBA interface names are without the prefix “I” in the interface name as used in the radio Management PIM Facilities.
2. UML attributes with readonly and readwrite map to CORBA attributes in CORBA interfaces.
3. UML attributes with configureproperty, queryproperty, and testproperty do not map to CORBA attributes in CORBA interfaces. Instead XML definitions are used that follow the Property types as defined in UML Profile for Component Framework::Application, Device Components::Properties section and UML Profile for Smart Antenna in the Chapter 7.
4. UML classes without operations that are not stereotyped and used for type definitions map to CORBA Struct stereotypes in the CORBA interfaces and modules. The parent classes do not get translated into CORBA types, instead the parent class attributes are added to the subclass in the CORBA definition.
5. UML <<datatype>> map to CORBA basic types. Primitive types are mapped to CORBA primitive types and primitive sequence types are mapped to CORBA Typedef of primitive sequence types.
6. UML exceptions and exception extensions map to CORBA exceptions. There is no specialization of exceptions in CORBA so the (UML Profile for Component Framework::Application and Device Components::BaseTypes) SystemException definition does not appear in the generated CORBA interfaces but all the specialization exceptions of SystemException are in the CORBA interfaces with the same attributes as defined for SystemException.
7. UML attributes that have a cardinality of many [*] map to a CORBA Typedef of sequence types.
8. UML operations and <<optional>> operations map to operations in the CORBA interfaces.
9. Transformations are only performed for concrete classes, not for template classes. Concrete classes that bind to template classes are used in the PSM.
10. For Interfaces that reference a component stereotype for a type, the “component” qualifier is removed from the name. For Example, FileManagerComponent would become FileManager as the type for the parameter or attribute.
11. UML attributes with constant stereotype map to CORBA constants in CORBA interfaces.
12. Basic types (e.g., Any, Object) map to CORBA types.
13. Object references map to the name of CORBA objects.

9.2 IDL Mapping

Table 9.1 - IDL Mapping

PIM Name	IDL FileName
SAControl	DfSWRadioSmartAntenna.idl
AlgorithmControl	DfSWRadioSmartAntenna.idl
RFControl	DfSWRadioSmartAntenna.idl
SynchronizationControl	DfSWRadioSmartAntenna.idl
Calibration	DfSWRadioSmartAntenna.idl
ChannelEstimation	DfSWRadioSmartAntenna.idl
SAAlgorithmDevice	DfSWRadioSmartAntenna.idl
Beamforming	DfSWRadioSmartAntenna.idl
DOAEstimation	DfSWRadioSmartAntenna.idl
SpaceTimeCoding	DfSWRadioSmartAntenna.idl
SpatialMultiplexing	DfSWRadioSmartAntenna.idl
Synchronization	DfSWRadioSmartAntenna.idl