Robotic Localization Service (RLS)

Version 1.1 with change bars

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/report_issue.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 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:

Business Modeling Specifications

Middleware Specifications

- **CORBA/IIOP**
- **Data Distribution Services**
- **Specialized CORBA**

IDL/Language Mapping Specifications

Modeling and Metadata Specifications

- **UML, MOF, CWM, XMI**
- **UML Profile**

Modernization Specifications

Platform Independent Model (PIM), Platform Specific Model (PSM), Interface Specifications

- **CORBAServices**
- **CORBAFacilities**

OMG Domain Specifications

CORBA Embedded Intelligence Specifications

CORBA 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/ report_issue.htm*.

1   Scope

This specification defines a localization (RoLo) service that can handle data and usages specific to use in robotics. It includes a platform independent model (PIM) as well as a mapping of this PIM to platform specific model (PSM) defined by C++. In addition, two informative annexes are provided for the filter condition functionality. The first defines a PSM by XML and the other shows naming rules.

2   Conformance

Any implementation or product claiming conformance to this specification shall support the following conditions:

- Implementations shall provide interfaces described in "Section 7.6, Interface Package."

- Implementations shall provide ability descriptors the necessary attribute definitions described in "Section 7.6, Interface Package."

- Data treated by implementations shall follow the structure described in "Section 7.3, Architecture Package" and data formats described in "Section 7.4, DataFormat Package." This does not mean that modules shall be able to treat every structure or formats described herein. However, every module shall support at least one of the common data formats and the relevant data structure.

- Implementations shall support the return codes described in "Section 7.2, Return Codes."

3   References

3.1   Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this specification. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply.

[ISO19103] International Organization for Standardization, Geographic information – Conceptual schema language, 2005

[ISO19107] International Organization for Standardization, Geographic information – Spatial schema, 2003

[ISO19111] International Organization for Standardization, Geographic information – Spatial referencing by coordinates, 2007

[ISO19115] International Organization for Standardization, Geographic information – Metadata, 2003

[PER] International Telecommunication Union Telecommunication Standardization Sector, Specification of Packed Encoding Rules (PER), ITU-T Rec. X.691 (2002) / ISO/IEC 8825-2:2002

[UML] Object Management Group, OMG Unified Modeling Language (OMG UML), Superstructure, Version 2.2, OMG document number formal/2009-02-02, 2009

3.2   Non-Normative References

[ISO/DIS19142] International Organization for Standardization, Geographic information – Web Feature Service, DIS, 2009

[ISO/DIS19143] International Organization for Standardization, Geographic information – Filter Encoding, DIS, 2009

[Noda2010] Itsuki Noda, Shuichi Nishio, Takashi Tsubouchi, Takeshi Sakamoto and Satoshi Tadokoro, "Mathematical Framework for Localization Information Coordinate Reference System for Robotics," in Proc. of Int. Workshop on Standards and Common Platform for Robotics 2010 (SCPR 2010), 2010.

[Wikipedia] Wikipedia, the free encyclopedia, http://www.wikipedia.org/

4 Terms and Definitions

Cartesian coordinate system	Coordinate system that gives the position of points relative to n mutually perpendicular axes. [ISO19111] Note that in this specification, in contrast to, there is no limitation to the number of dimensions. [ISO19111]
Coordinate reference system (CRS)	Coordinate system that is related to the real world by a datum. [ISO19111]
Coordinate system (CS)	Set of mathematical rules for specifying how coordinates are to be assigned to points. [ISO19111]
Coordinate value	N-tuple of scalars assigned with respect to a coordinate system. In this specification, every coordinate value shall be associated with a single coordinate reference system. Note that, there exists no uncertainty with a coordinate value; error through the measurement process shall be represented by 'error' values elsewhere.
Covariance	Covariance is a measure of how much two variables change together (variance is a special case of the covariance when the two variables are identical). If two variables tend to vary together (that is, when one of them is above its expected value, then the other variable tends to be above its expected value too), then the covariance between the two variables will be positive. On the other hand, if one of them tends to be above its expected value when the other variable is below its expected value, then the covariance between the two variables will be negative. [Wikipedia]
Datum	Parameter or set of parameters that define the position of the origin, the scale, and the orientation of a coordinate reference system. [ISO19111] More specifically, a datum is a mathematical system that defines the mapping from a space defined by coordinate system to a certain phenomenon space of interest, mostly in the real world.
Geodetic coordinate system	Coordinate system in which position is specified by geodetic latitude, geodetic longitude, and (in the three-dimensional case) ellipsoidal height associated with one or more geographic coordinate reference systems. [ISO19111]
Geographic(al) Information System (GIS)	Information system for storing, analyzing, managing, or displaying various data in a way associated with location data. The location data used in GIS is in most cases 2 or 3 dimensional position on the earth.

Kalman filter	Kalman filter is an efficient recursive filter that estimates the state of a linear dynamic system from a series of noisy measurements. It is used in a wide range of engineering applications from radar to computer vision, and is an important topic in control theory and control systems engineering. Together with the linear-quadratic regulator (LQR), the Kalman filter solves the linear-quadratic-Gaussian control problem (LQG). The Kalman filter, the linear-quadratic regulator, and the linear-quadratic-Gaussian controller are solutions to what probably are the most fundamental problems in control theory. [Wikipedia]
Localization	Action of locating some physical entities through analysis of sensing data. The word "locate" here may include not only measuring the position in the spatio-temporal space but also may include additional information such as identity, heading orientation, or pose information of the target entity, measurement error estimation, or measurement time.
Normal distribution	A continuous probability distribution described by the following probability density function: $$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$ A normal distribution is also called a Gaussian distribution. [Wikipedia]
Particle, particle set	A particle is a word used to denote a single sample obtained through random sampling algorithms such as Monte Carlo method. A particle set is a set of samples obtained through some sampling or estimation algorithms. robotics, particles and particle sets are often used to represent distributions obtained from estimation algorithms such as sequential Monte Carlo method or CONDENSATION (**Con**ditional **Dens**ity Propag**ation**) algorithm.
Physical entity	The target to be localized such as robots, humans, or other objects.
Polar coordinate system	Two-dimensional coordinate system in which position is specified by distance and direction from the origin. [ISO19111] In this specification, three-dimensional coordinate system (spherical coordinate system) or n-dimensional coordinate system may also be called as polar coordinate system.
Data Instance	A 'data instance' is a RoLo data or its subcomponent such as a RoLo element, a RoLo position, a RoLo symbolic position, or a GM_Position object.
Implicit (Type) Specification	When a structure embedding some data instance holds a type specification for those data instances, those data instances are described to have an "implicit type specification." For example, a RoLo data is implicitly associated with a RoLo data specification when the data is passed through a RoLo stream that holds a RoLo data specification defined in its ability description.

Explicit (Type) Specification	A data instance is said to have an "explicit type specification" if a reference to corresponding specification is provided in its attribute. For example, when a RoLo data has a reference to RoLo data specification as its 'spec' attribute, the RoLo data is said to have an explicit type specification.
Type Specification	A "type specification" of a data instance is either an implicit type specification or an explicit type specification of an instance.
Incomplete (Type) Specification	A type specification is called "incomplete" when it includes one or more "don't-care" elements.
Complete (Type) Specification	A type specification is called "complete" when it does not include any "don't-care" element.
Consistent Type Specifications	Two type specifications are called "consistent" when the two specifications own the same structure and each corresponding parts of them is the same or have a base-derivation (generalized-specialized) relation with each other, or when one of the corresponding parts are specified as "don't-care."
Unified (Type) Specification	A "unified type specification" of a data instance is the result of unification of all the type specifications associated with the data instance. The type specifications to be unified shall be consistent. The unification of two type specifications is done by the following operation. For each part of the type specifications, do: • When both of the corresponding type specifications are "don't-care," use "don't-care." • When one of part of the two specifications is "don't-care," use the corresponding part from another specification. • When both of the corresponding type specifications are not "don't-care," use the one that is much specialized.

5   Symbols

x, y, z   Cartesian coordinate

r, θ, φ   spherical coordinate

φ, λ, *h*   geodetic coordinate (latitude, longitude, height)

α, β, γ   orientation

x, y, z   a fixed Cartesian coordinate system

6 Additional Information

6.1 Background

This specification defines a localization service that can handle data and usages specific to use in robotics. It includes a platform-independent model (PIM) as well as a mapping of this PIM to platform-specific models (PSM) defined by C++.

Location information is a crucial factor in providing robotic services of every kind. Typically, a robotic system is defined as an apparatus equipped with the function of interacting with physical entities in the environment. Navigation, manipulation, and human-robot interaction are typical features that require physical interaction with the environment, which distinguish a robotic system from information appliances. On performing such tasks, robots require geometric association between physical entities of interest and the robot itself for implementing and/or performing the given service scenario. Besides these examples, the number of location-based robotic tasks is continuously increasing as personal or service robot fields gradually expand, from controlled, stable factory environments to indeterminate, uncertain daily environments. However, currently there exists no standard means to represent the necessary location-related information in robotics, nor any common interface for constructing localization related software modules.

Note – In the context of this specification, the word "**localization**" means "*to locate some physical entities through analysis of sensor data*," consistent with the common use of this term in robotics. Here the word "**locate**" may include not only measuring the position in the spatio-temporal space, but also heading orientation or pose information of the entity, or additional information such as error estimation or time of measurement. Also, the word "**physical entity**" (or "entity" in short) is used to describe the target to be localized, including robots, humans, or other objects.

Geographic Information System (GIS) is one of the most popular and established systems that treats location information. Many spatio-temporal location related specifications have been standardized in the International Organization for Standardization (ISO/TC211). Versatile production services already exist based on these standards such as driving navigation systems or resource databases. However, current GIS specifications are not powerful enough to represent or treat information required in the field of robotics.

Although localization is still one of the main research topics in the field of robotics, the fundamental methodology and necessary elements are becoming established. Standardizing localization result representation and related interfaces in a generic form, independent to specific algorithms or equipment, are significant for decreasing costs and accelerating the market growth of robotic services. Moreover, clarifying what types of information are required in the field of robotics shall be useful for equipment vendors such as sensor manufacturers.

This document presents a new framework for robotic localization (**RoLo**) (i.e., representing and treating location information specific to robotic usage). Notions and items necessary for treating location information in robotic usage are reorganized and rearranged, in a generic form independent to specific algorithms or types of robotic services. This was done through extensive surveys and case studies on current and ongoing robotic products and researches. Based on the widespread GIS standard, a new specification for services is proposed.
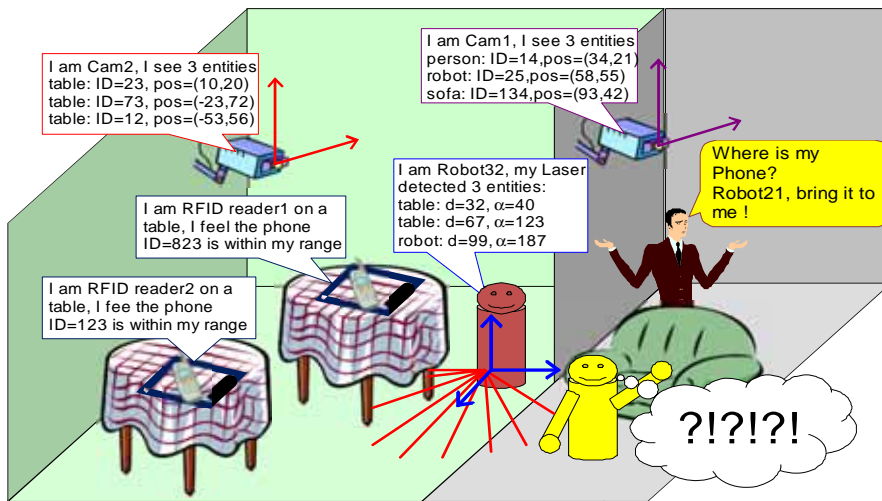
Figure 6.1 - Example of a typical robotic service situation requiring localization of an entity

Figure 6.1 illustrates a typical robotic service situation where localization of various entities is required. Here, a robot in service needs to obtain the location of a cellular phone, utilizing information from various robotic entities in the environment. These robotic entities have the ability to estimate the location of the entities within their sensing range. Thus, the problem here is to aggregate the location estimations from the robotic entities, and to localize the cellular phone in target. However, this example also shows several factors that make the localization service in robotics a difficult, challenging issue.

- **Some sensors only provide partial location information.** For example, the camera sensor can only provide 2D information, and RF tag reader can only provide proximity information.

- **Sensor outputs are not always correct.** Sometimes, they might measure two or more entities as a single object, or even miss it. This erroneous report occurs frequently when sensors are used in the uncontrolled daily environment. To tackle this erroneous situation, sensor outputs are usually treated to be probabilistic, with error estimation information.

- **Matching observations between different sensors require efforts.** Imagine you are viewing two photographs of a crowded street corner, taken from different angles but on the same instant. The issue here is to match every single person in one photograph to another. This is much more difficult when matching the observed entities from the wall camera and the output from the laser range scanner installed in the blue robot, as these two sensors measure different aspects of objects. This issue, the *identity association problem*, happens every time multiple sensors are used. In other words, you are not always sure about the identity of the entity sensed. Thus, identity information shall also be treated to be probabilistic.

As can be seen from the examples in Figure 6.1, operations in robotics require a much more detailed representation of location information. Still, interoperability with the current GIS systems shall be supported. In this specification, a new framework for representing and treating location information suitable for robotic use is defined by extending existing GIS specifications. Using the GIS specification as a basis of the specification will make it easy for robots to interconnect with existing GIS-based systems and utilize existing geographic datasets. This will also ease the use of this specification in the emerging fields of next-generation GIS systems, sensor network systems, or location based systems where advanced positioning methods and complex data processing similar to robotics usage is required. Figure 6.2 illustrates the existing GIS standards that are related with this specification.
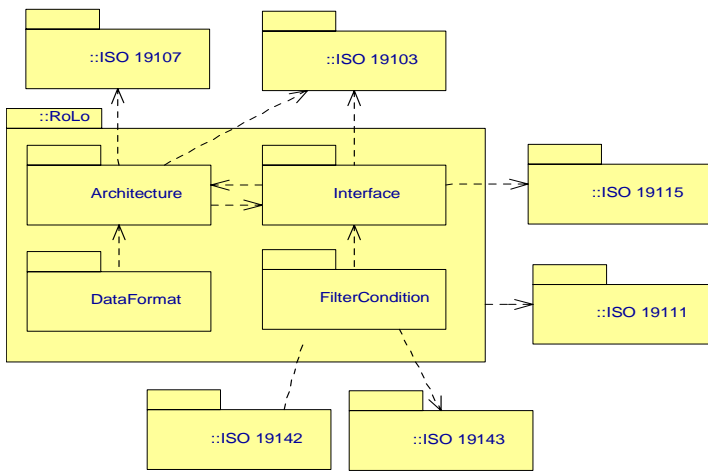
Figure 6.2 - Relation of Robotic Localization Service specification with existing GIS specifications

To fulfill the requirements for robotic localization, the following items are defined in the PIM, some part as an extension to existing GIS specifications.

- Architecture package - Data architecture for representing structures and accompanying operations for representing information necessary for robotics usage. These include coordinate system / coordinate reference system definitions for treating essential information such as pose or identity information, or structures for representing error estimation.

- DataFormat package - Data formats for formatting and exchanging resulting localization data.

- Interface package - Service interface for treating resulting localization data. This includes advanced facilities that will be a basis for dynamically exchanging or negotiating module functionality information.

6.2 Acknowledgements

6.2.1 Submitters

The initial submissions that this specification is based on were submitted by the following people:

- Kyuseo Han, Electronics and Telecommunications Research Institute (ETRI)
- Yeonho Kim, Samsung Electronics Co., Ltd.
- Shuichi Nishio, Japan Robot Association (JARA) / Advanced Research Institute International (ATR)

6.2.2 Submitting Organizations

The following organizations were responsible for the initial submission that this specification is based on:

- Electronics and Telecommunications Research Institute (ETRI)
- Japan Robot Association (JARA)
- Samsung Electronics Co., Ltd.

6.2.3   Supporters

The following people supported parts of this specification:

- Su-Young Chi, Electronics and Telecommunications Research Institute
- Yun Koo Chung, Electronics and Telecommunications Research Institute
- Miwako Doi, Toshiba Corporation
- Kenjirou Fujii, Hitachi Industrial Equipment Systems Co., Ltd.
- Yoshimasa Hata, Japan Robot Association
- Yasuo Hayashibara, Chiba Institute of Technology
- Ryota Hiura, Mitsubishi Heavy Industries, Ltd.
- Toshio Hori, National Institute of Advanced Industrial Science and Technology
- Masato Iehara, Mitsubishi Heavy Industries, Ltd.
- Wataru Inamura, IHI Corporation
- Jaeyeong Lee, Electronics and Telecommunications Research Institute
- Takahide Kanehara, Yaskawa Electric Corporation
- Tetsuo Kotoku, National Institute of Advanced Industrial Science and Technology
- Makoto Mizukawa, Shibaura Institute of Technology
- Kouji Murakami, Kyushu University
- Yoshisada Nagasaka, National Agriculture and Food Research Organization
- Itsuki Noda, National Institute of Advanced Industrial Science and Technology
- Kohtaro Ohba, National Institute of Advanced Industrial Science and Technology
- Fumio Ozaki, Toshiba Corporation
- Takeshi Sakamoto, Technologic Arts Incorporated
- Takashi Suehiro, National Institute of Advanced Industrial Science and Technology
- Tetsuo Tomizawa, National Institute of Advanced Industrial Science and Technology
- Takashi Tsubouchi, University of Tsukuba
- Tomoki Yamashita, Maekawa MFG Co. Ltd.
- Masayoshi Yokomachi, New Energy and Industrial Technology Development Organization
- Wonpil Yu, Electronics and Telecommunications Research Institute

6.2.4   Supporting Organizations

The following organizations supported parts of this specification:

- Hitachi, Ltd.
- National Institute of Advanced Industrial Science and Technology (AIST)
- New Energy and Industrial Technology Development Organization (NEDO)
- Shibaura Institute of Technology
- Technologic Arts Incorporated
- University of Tsukuba

7 Platform Independent Model

The Platform Independent Model (PIM) consists of three parts:

1. Architecture package - The architecture package defines a new framework for representing location information required in the field of robotics. See "Architecture Package" on page 10.

2. DataFormat package - The data format package defines how the defined data is represented for exchange among RoLo modules. See "DataFormat Package" on page 38.

3. Interface package - The interface package defines an API for data passing and configuration of RoLo modules. See "Filter Condition Package" on page 41.

7.1 Format and Conventions

7.1.1 Class

Classes described in this PIM are documented using tables of the following format.

Table xx: <class name>

Description: <description>				
Derived From: <parent class>				
Attributes				
<attribute name>	<attribute type>	<obligation>	<occurrence>	<description>
…	…			…
Operations				
<operation name>		<description>		
<direction>	<parameter name>	<parameter type>		<description>
…	…	…		…

Note that derived attributes or operations are not described explicitly. Also, as the type of return code for every operation in this specification is Returncode_t, which is defined in "Return Codes" on page 10, this is omitted in the description table.

The 'obligation' and 'occurrence' are defined as follows.

Obligation

- **M (mandatory):** This attribute shall always be supplied.

- **O (optional):** This attribute may be supplied.

- **C (conditional):** This attribute shall be supplied under a condition. The condition is given as a part of the attribute description.

Occurrence

The occurrence column indicates the maximum number of occurrences of the attribute values that are permissible. The following denotes special meanings.

- **N:** No upper limit in the number of occurrences.

- **ord:** The appearance of the attribute values shall be ordered.

- **unq:** The appeared attribute values shall be unique.

7.1.2   Enumeration

Enumerations are documented as follows:

Table xx: <enumeration name>

<constant name>	<description>
…	…

7.2   Return Codes

At the PIM level, we have modeled errors as operation return codes typed **Returncode_t**. Each PSM may map these to either return codes or exceptions. The complete list of return codes is indicated below.

Table 7.1 - Returncode_t enumeration

OK	Successful return
ERROR	Generic, unspecified error
BAD_PARAMETER	Illegal parameter value
UNSUPPORTED_PARAMETER	Unsupported parameter
UNSUPPORTED_OPERATION	Unsupported operation
TIMEOUT	The operation timed out

7.3   Architecture Package

Modern robotic algorithms related to localization require not only simple spatial positioning information; generally, various types of information related to spatial position are also required. To obtain precise results, measurement time, and error estimation is crucial, especially when integrating measurements from multiple sensors. For robotics usage, complex spatial positioning such as pose information is also important. When sensors in use can perform measurements of multiple entities at once, identity information is also necessary to distinguish and associate measurements. As such, there is a variety of other information to be expressed in combination with simple spatial positioning. To make various robotic services treat and process this versatile information easily and effectively, we represent this heterogeneous information under a common, unified framework.

In this sub clause, we discuss a new framework for representing location information required in the field of robotics, by extending existing GIS specifications. First, we define a new framework for representing position and pose information. Next, three types of information required in robotics usage are defined, and lastly, a generic framework for representing structured robotic localization results (RoLo architecture) is defined.

Note that, although the ISO 19111 specification assumes every CS to be 2 or 3 dimensional [ISO19111], in this specification, we do not assume any limitation on the number of dimensions on any coordinate systems. This is to enable representation of complex data such as feature points defined over multi-dimensional space. Also note that this does not violate the ISO 19111 standard where no formal limitation is specified on the number of dimensions. One issue is how to treat the attribute bounded to specific features in the real space such as axisDirection (type CS_AxisDirection) in CS_CoordinateSystemAxis that is a mandatory attribute, and where the type is defined as a finite enumeration of direction names such as 'north' or 'south.' It is clear that these values are not suitable for some robotics usage such as for relative or mobile coordinate reference systems. We thus recommend that implementers and users of this specification simply ignore this attribute and set this value as the first element in the enumeration, 'north,' if necessary. This is a safe solution as we cannot expect GIS systems to treat data based on this specification correctly; we only expect data from GIS systems to be treated on systems based on this robotic specification.

7.3.1   Pose Representation

In this sub clause, we first extend the notion of coordinate reference system [ISO19111] for representing pose information. The simplest way for representing pose information is to define a special CS and CRS that represents pose. However, pose information, in general, can only be considered and interpreted in the context of position information. See [Noda2010] for detailed discussion. Therefore, here we represent pose information in combination of position information. In order to realize this representation is consistent with ISO 19111, we define a new subclass of CRS for representing a set of position and pose information while preserving the relation to the original position-only CRS.
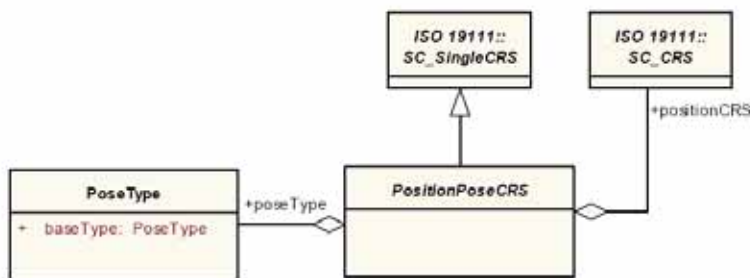
Figure 7.1 - Pose Information

Table 7.2 - PositionType class

Description: Class for indicating how pose is represented				
Derived From: (none)				
Attributes:				
baseType	PoseType	O	1	Reference to a RoLo pose type that is the base type of this RoLo pose type. This attribute is used to represent hierarchical relationships among RoLo pose types.

Table 7.3 - PositionPoseCRS class

Description: A CRS for denoting a combination of position and pose information. The attributes derived from: SC_SingleCRS shows the base CRS for representing the pose information. Note that the dimension of the coordinate values based on this CRS is not equal to the dimension of this CRS and varies based on how the pose information is represented. For example, when this CRS is 3 dimensional, the coordinate value's dimension may be 3 for Euler angle representations and 4 for quarternion representations.				
Derived From: SC_SingleCRS [ISO19111]				
Attributes:				
poseType	PoseType	M	1	Reference to a RoLo pose type that shows how the pose information is represented.
positionCRS	SC_CRS [ISO19111]	M	1	Reference to a CRS that indicates which CRS the position information is based on.

7.3.1.1 Pose Representation Type

We also define PoseType instances of common pose representations as in Figure 7.2.
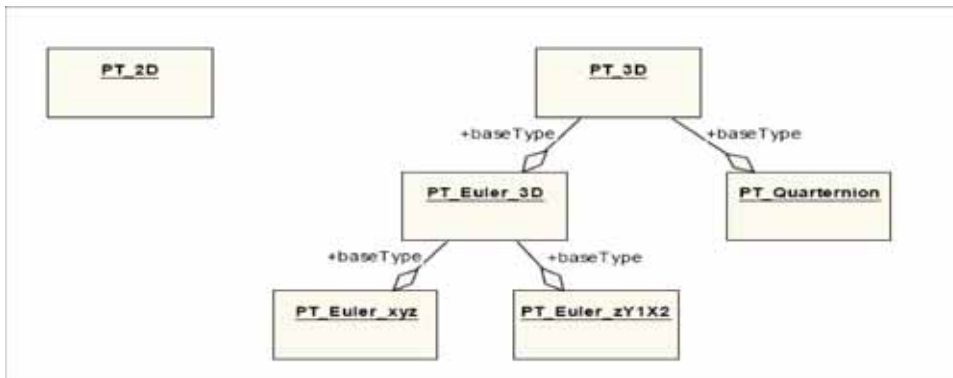
Figure 7.2 - Pose representation types

Generally, a 3 by 3 matrix is commonly used in robotics to calculate consecutive rotations of a coordinate system or to specify the orientation of a coordinate system respective to a reference coordinate system. We will call this pose representation style as PT_3x3matrix.

Besides, there is a common representation called Euler angles that specify the orientation of a coordinate system by a sequence of three rotations that take place about an axis of the coordinate system. Depending on the order of sequential rotations of the two coordinate systems, the Euler angle representation can be defined in several ways. To specify the rotation order of the coordinate system, a fixed right-hand Cartesian coordinate system is denoted in lower case (x, y, z). Also we denote the right-hand Cartesian coordinate system after nth rotation as (Xn, Yn, Zn). We name the Euler angle representations by the order of rotation axis.

For example, PT_Euler_xyz is defined as follows:

1. Rotate the coordinate system about the x-axis by a, as shown in Figure 7.3 (a).

2. Rotate the coordinate system about the fixed y-axis by ß, as shown in Figure 7.3 (b).

3. Rotate the coordinate system about the fixed z-axis by ?, as shown in Figure 7.3 (c).
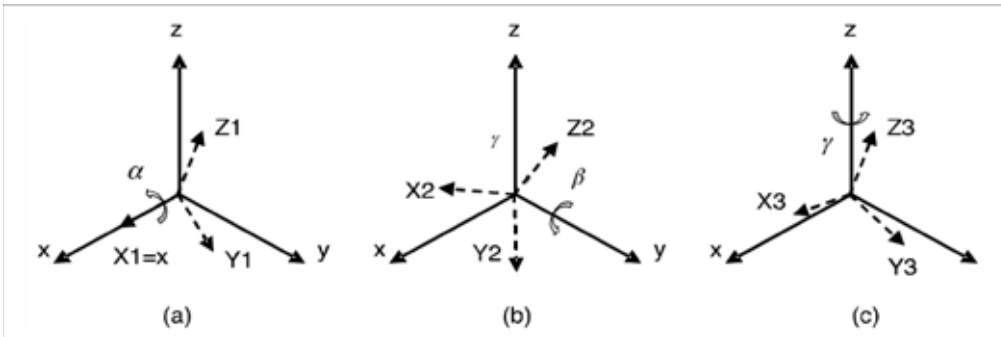
Figure 7.3 - Three sequential rotations for the PT_Euler_xyz pose representation

Another example, PT_Euler_zY1X2 is defined as follows:

1. Rotate the coordinate system about the z-axis by a as shown in Figure 7.4 (a). The resulting coordinate system is denoted as (X1, Y1, Z1).

2. Rotate the coordinate system about the Y1-axis by ß as shown in Figure 7.4 (b). The resulting coordinate system is denoted as (X2, Y2, Z2).

3. Rotate the coordinate system about the X2-axis by ?, as shown in Figure 7.4 (c).
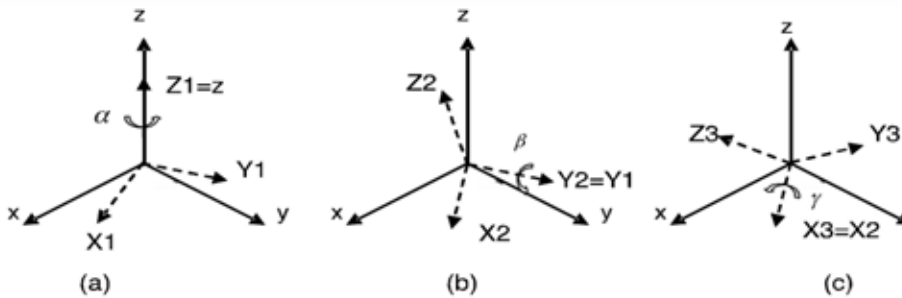
Figure 7.4 - Three sequential rotations for the PT_Euler_zY1X2 pose representation

7.3.2   Relative Coordinate Reference Systems

In this sub clause, relative coordinate reference systems are defined that may lack fixed relation with the earth or users have no interest in referencing them to other coordinate reference systems. We categorize relative coordinate reference systems in two types, static and dynamic. A coordinate reference system on mobile platforms, mobile coordinate reference system, is defined as a dynamic relative coordinate reference system. That is, the relation with other coordinate reference systems may change by time.

The GIS standard on spatial reference system [ISO 19111] allows the definition and use of such relative and mobile coordinate reference systems. However, there is no specific model or description on these systems. As these systems are quite commonly used in the field of robotics, here we explicitly define structures and operations specific to these coordinate reference systems. Although we only define coordinate reference systems based on two coordinate systems of frequent usage, CS_CartesianCS and CS_PolarCS, users may define derivatives of relative or mobile coordinate reference systems based on the coordinate system of their interest.
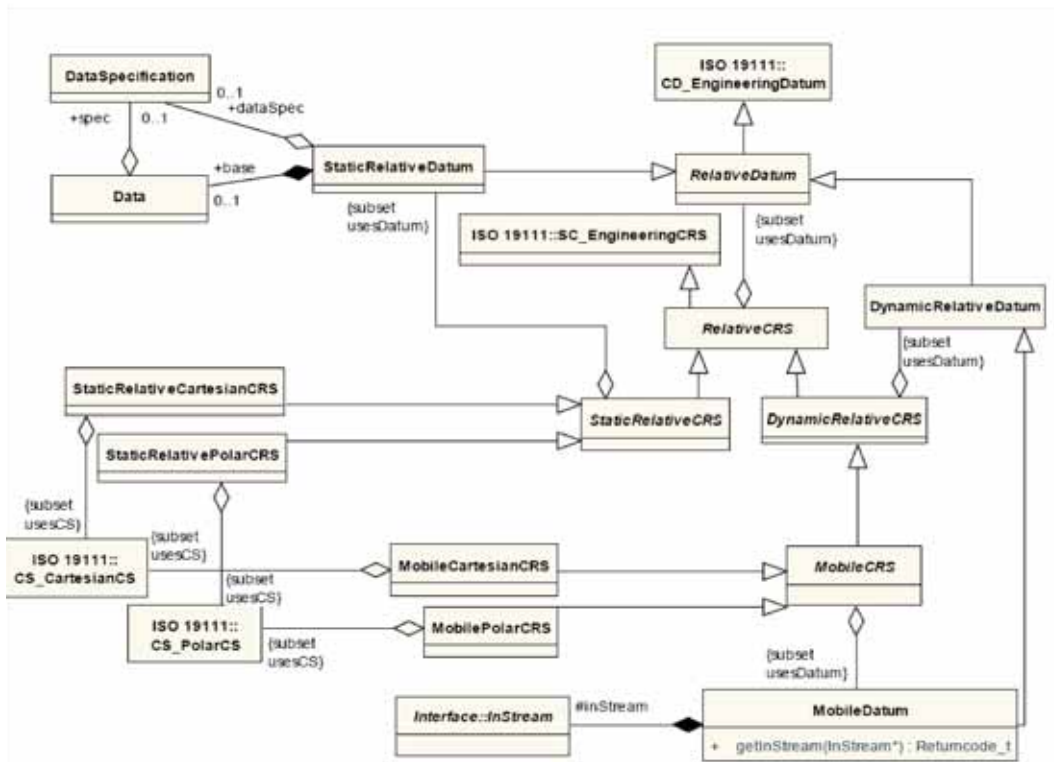
Figure 7.5 - Relative and Mobile coordinate reference system

Table 7.4 - RelativeCRS class

Description: Base abstract class for representing relative coordinate reference systems.
Derived From: SC_EngineeringCRS [ISO19111]
Note – Values for the attribute 'UsesDatum' that are derived from parent class shall be limited to instances of RelativeDatum or its inherited classes.

Table 7.5 - RelativeDatum class

Description: Abstract class for representing datum for relative coordinate reference systems.
Derived From: CD_EngineeringDatum [ISO19111]

Table 7.6 - StaticRelativeCRS class

Description: Abstract class for representing relative coordinate reference systems that have static relation with other CRS(s).
Derived From: RelativeCRS
Note – Values for the attribute 'usesDatum' that are derived from parent class shall be limited to instances of StaticRelativeDatum or its inherited classes.

Table 7.7 - StaticRelativeCartesianCRS class

Description: Static relative coordinate reference systems based on Cartesian coordinate system.	
Derived From: StaticRelativeCRS	
Note – Values for the attribute 'usesCS' that are derived from parent class shall be limited to instances of CS_CartesianCS [ISO19111] or its inherited classes.	

Table 7.8 - StaticRelativePolarCRS class

Description: Static relative coordinate reference systems based on polar coordinate system.	
Derived From: StaticRelativeCRS	
Note – Values for the attribute 'usesCS' that are derived from parent class shall be limited to instances of CS_PolarCS [ISO19111] or its inherited classes.	

Table 7.9 - StaticRelativeDatum class

Description: Datum for static relative coordinate reference system				
Derived From: RelativeDatum				
Attributes:				
dataSpec	DataSpecification	O	1	A RoLo data specification indicating allowed structure for the 'base' attribute. If the coordinate reference system in target holds no relation with other coordinate reference systems, this may be omitted.
base	Data	O	1	A RoLo data for determining relation to other coordinate reference system. Typically, this data includes spatial position for origin and pose for axis direction. If no relation with other coordinate reference systems is required, this may be omitted.

Table 7.10 - DynamicRelativeCRS class

Description: Abstract base class for representing dynamic relative coordinate reference systems.	
Derived From: RelativeCRS	
Note – Values for the attribute 'usesDatum' that are derived from parent class shall be limited to instances of DynamicRelativeDatum or its inherited classes.	

Table 7.11 - DynamicRelativeDatum class

Description: Datum for dynamic relative coordinate reference system	
Derived From: RelativeDatum	

Table 7.12 - MobileCRS class

Description: Abstract base class for representing mobile coordinate reference systems.
Derived From: DynamicRelativeCRS
Note – Values for the attribute 'usesDatum' that are derived from parent class shall be limited to instances of MobileDatum or its inherited classes.

Table 7.13 - MobileCartesianCRS class

Description: Mobile coordinate reference systems based on Cartesian coordinate system.
Derived From: MobileCRS
Note – Values for the attribute 'usesCS' that are derived from parent class shall be limited to instances of CS_CartesianCS [ISO19111] or its inherited classes.

Table 7.14 - MobilePolarCRS class

Description: Mobile coordinate reference system based on polar coordinate system.

Table 7.15 - MobileDatum class

Description	: Datum for mobile coordinate reference systems. This datum holds a RoLo input stream that is used to obtain positional information for determining the relation between the mobile coordinate reference system in target and another coordinate reference system. Users shall connect a RoLo output stream to this input stream, or shall supply positional information directly by the 'setData' method of this input stream. For example, if the mobile coordinate information for determining axis directions may be supplied. However, some transformation algorithms require more complicated information such as measurement time or error information. The necessary information required can be determined by the ability description of the input stream.			
Derived From: DynamicRelativeDatum				
Attributes:				
inStream (protected)	InStream (RoLo::Interface)	M	1	Input stream for obtaining base position.
Operations:				
getInStream	Returns the input stream in use.			
out / inStream	InStream (RoLo::Interface)		InStream instance used in this datum.	
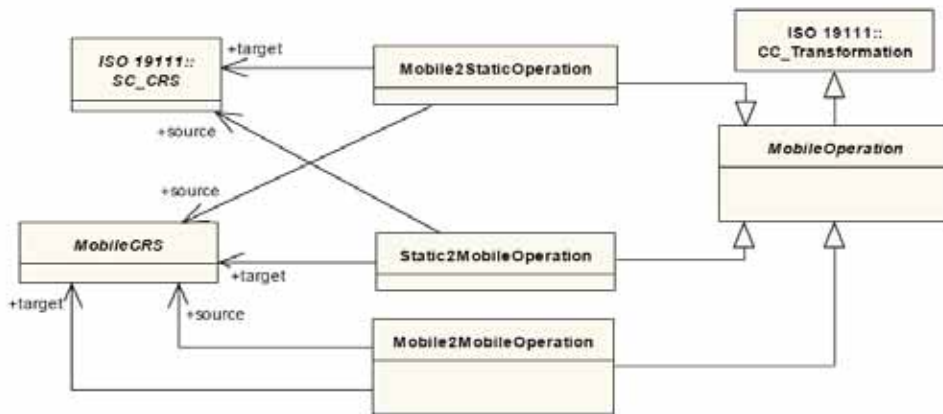
Figure 7.6 - Mobile CRS Operations

Table 7.16 - MobileOperation class

Description: Abstract base class for operations between mobile coordinate reference system and other coordinate reference systems.
Derived From: CC_Transformation [ISO19111]

Table 7.17 - Mobile2StaticOperation class

Description: Transformation operation from mobile coordinate reference systems to other static, non-mobile coordinate reference.				
Derived From: MobileOperation				
Attributes:				
source	MobileCRS	M	1	The source mobile coordinate reference system
target	SC_CRS [ISO19111]	M	1	The target coordinate reference system
Note – Values for the attribute 'target' shall not be an instance of DynamicRelativeCRS or its inherited classes.				

Table 7.18 - Static2MobileOperation class

Description: Transformation operation from other static non-mobile coordinate reference systems to mobile coordinate reference systems.				
Derived From: MobileOperation				
Attributes:				
source	SC_CRS [ISO19111]	M	1	The source coordinate reference system
target	MobileCRS	M	1	The target mobile coordinate reference system
Note – Values for the attribute 'source' shall not be an instance of DynamicRelativeCRS or its inherited classes.				

Table 7.19 - Mobile2MobileOperation class

Description: Transformation operation between mobile coordinate reference systems				
Derived From: MobileOperation				
Attributes:				
source	MobileCRS	M	1	The source mobile coordinate reference system
target	MobileCRS	M	1	The target mobile coordinate reference system

7.3.3   Identity Information

Identity (ID), which is assigned for each localized target, can also be treated as a value on some coordinate reference system. For example, MAC addresses used in Ethernet communication protocols can be represented as a coordinate value on a two-dimensional coordinate system, vendor code, and vendor-dependent code. Electric Product Code (EPC) or ucode, used for identifying RF tags, is another example of identification systems defined by a multi-dimensional coordinate system. There also exist some ID systems, such as family names, that are usually not explicitly defined over some mathematical structure.

In general, each sensor holds its own ID system and each entity observed is assigned an ID from this local ID system. This is because, at least on the initial stage, there are no means to assign the observed entity a global ID. Thus, when multiple sensors are in use, there exist multiple local ID systems independent to each other, and it becomes necessary to properly manage and integrate these ID systems. Resolving the bindings between each local ID system is called the ID association problem, and is one of the major research issues in the robotic localization field. Also, as we saw in the overview, ID assignments are probabilistic, just like other location information.

Under these considerations, here we define coordinate reference systems and related structures for representing identity information. Here, two coordinate reference systems and accompanying coordinate systems are defined, for identity systems that are represented in numerical values and symbolic values. The actual coordinate value holding structure in GIS standard [ISO19107] only allows numeric values as coordinate value elements. Thus, similar structures in use with symbolic values are also defined.

Note that, operations on identity information (such as conversion from numeric ID to symbolic ID or mapping between different ID systems) can be constructed using CC_CoordinateOperation or relevant classes specified in GIS standard [ISO19111]. This is because the identity information defined here is represented by using derived classes from GIS coordinate systems and coordinate reference systems.
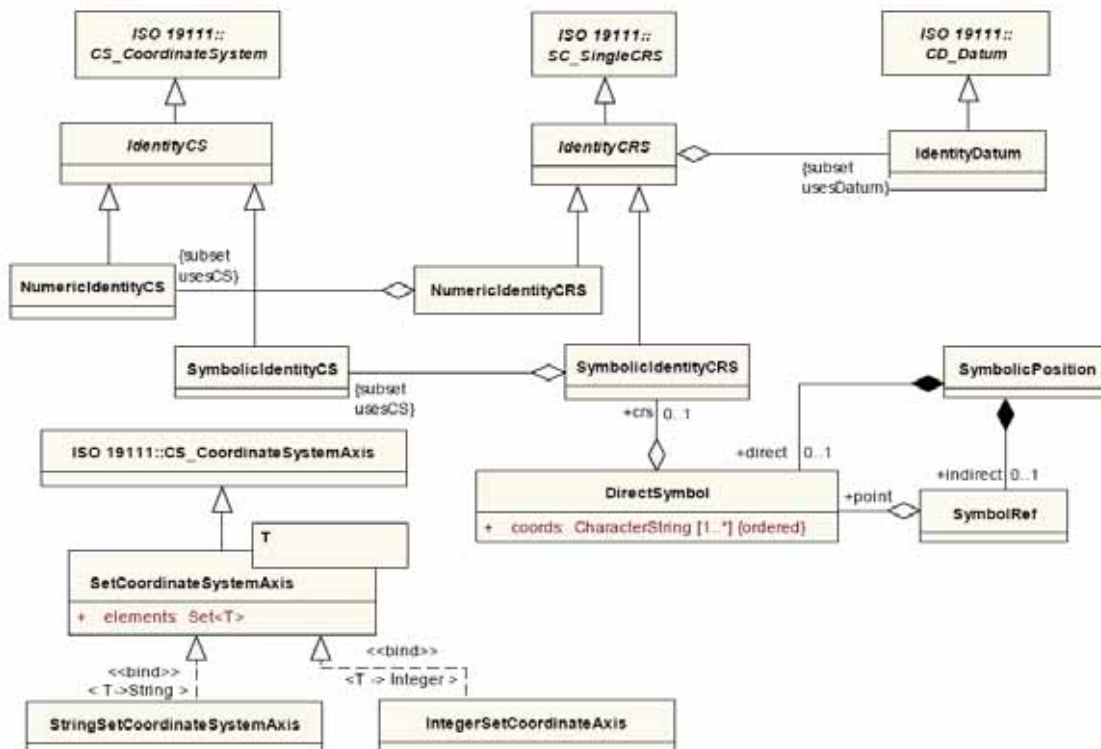
Figure 7.7 - Identity Information

Table 7.20 - IdentityCSclass

Description: Base abstract class for representing coordinate systems for identity information
Derived From: CS_CoordinateSystem [ISO19111]

Table 7.21 - NumericIdentityCSclass

Description: Coordinate system for identity information, where each axis is defined over numerical values.
Derived From: IdentityCS

Table 7.22 - SymbolicIdentityCSclass

Description: Coordinate system for identity information, where each axis is defined over a set of symbolic values.
Derived From: IdentityCS

Table 7.23 - IdentityDatumclass

Description: Datum for identity coordinate reference systems	
Derived From: CD_Datum [ISO19111]	

Table 7.24 - IdentityCRSclass

Description: Base abstract class for representing coordinate reference systems for identity information	
Derived From: SC_SingleCRS [ISO19111]	
Note – Values for the attribute 'usesDatum' that are derived from parent class shall be limited to instances of IdentityDatum or its inherited classes.	

Table 7.25 - NumericIdentityCRSclass

Description: Coordinate reference system for identity information, where each axis is defined over numerical values.	
Derived From: IdentityCRS	
Note – Values for the attribute 'usesCS' that are derived from parent class shall be limited to instances of NumericIdentityCS or its inherited classes.	

Table 7.26 - SymbolicIdentityCRSclass

Description: Coordinate reference system for identity information, where each axis is defined over a set of symbolic values.	
Derived From: IdentityCRS	
Note – Values for the attribute 'usesCS' that are derived from parent class shall be limited to instances of SymbolicIdentityCS or its inherited classes.	

Table 7.27 - DirectSymbol class

Description: Class for holding symbolic identity information				
Derived From: none				
Attributes:				
coords	CharacterString	M	N ord	Values for each of the coordinate system axis
crs	SymbolicIdentityCRS	O	1	Reference to the coordinate reference system this data belongs to

Table 7.28 - SymbolRef class

Description: Data holder for a reference to DirectSymbol				
Derived From: none				
Attributes:				
point	DirectSymbol	M	1	Reference to the target DirectSymbol class instance

Table 7.29 - SymbolicPosition class

Description: Union of DirectSymbol and SymbolRef. This class is used as a data holder for accessing symbolic information transparently, whether it is directly held or indirectly referenced.				
Derived From: none				
Attributes:				
direct	DirectSymbol	C	1	Symbolic identity data
indirect	SymbolRef	C	1	Reference to symbolic identity data
Condition: Either one of the elements shall be contained.				

Table 7.30 - SetCoordinateSystemAxis class

Description: CoordinateSystemAxis where the domain of axis values are represented by a finite set.				
Derived From: CS_CoordinateSystemAxis [ISO19111]				
Attributes:				
elements	Set<T>	M	1	Elements of the domain

Table 7.31 - StringSetCoordinateSystemAxisclass

Description: SetCoordinateSystemAxis with String values
Derived From: SetCoordinateSystemAxis

Table 7.32 - IntegerSetCoordinateSystemAxisclass

Description: SetCoordinateSystemAxis with Integer values
Derived From: SetCoordinateSystemAxis

7.3.4 Error Information

Every sensing system in the real world cannot avoid having measurement error. As such, it is essential to know the reliability or deviation of measurements for performing localization and for utilizing the resulting estimation. Error information plays an important role in robotic operations. In GIS specifications, the only error concerned is the expected reliability of inter-coordinate transformation. However, complex and detailed error descriptions are required in modern localization methods. Thus, here we define additional structures for representing and operating on error information.

7.3.4.1 RoLo Error Type

Similar to the relation of coordinate reference system and the position in the traditional GIS systems, we here define RoLo error types for describing the nature of error information. Every RoLo error holds a reference to an error type (either implicitly or explicitly; see Table 7.83), which indicates how this error is represented. This means that the same error data can be represented in a different manner. Thus, operations for transforming between different error types are defined.

RoLo error types may also be structured for a hierarchy. Just as the normal class inheritance relationships, often error types may be related to each other. For example, a linear mixture model distribution is one limited form of general mixture model where models mixture is performed through linear operations. Here the hierarchy of RoLo error types is specified by inter-object relationships, and not by inter-class relationships. This is to be consistent with other specification data types such as coordinate reference system, coordinate system, or RoLo data specification. Figure 7.8 shows some RoLo error types and their relationships corresponding to the RoLo error classes defined afterwards.
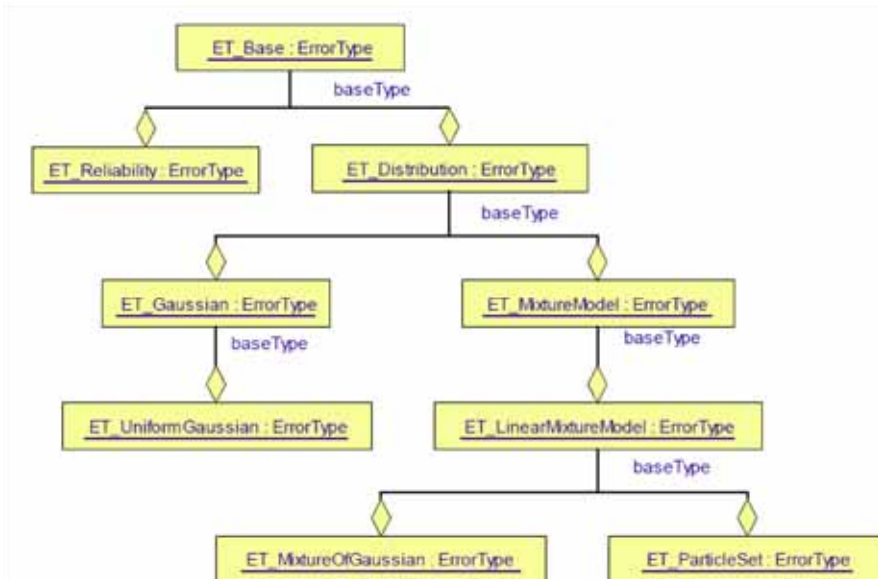
Figure 7.8 - Hierarchy of RoLo Error Types

Note that error information in the context of localization cannot exist solely by itself. Error information is an attribute to the location value. Thus, there exist two types of operation on error information in general. 1) Change in error representation type, and 2) change in the coordinate system the target location value is based on. The former operation is described in this sub clause, and the latter is described later with the description of RoLo data specification.
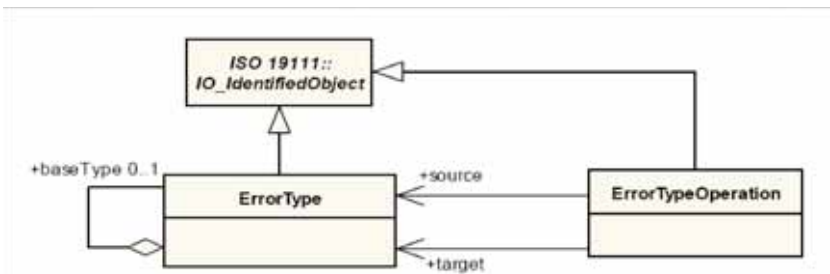
Figure 7.9 - RoLo Error Types

Table 7.33 - ErrorType class

Description: Class for representing RoLo error types				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes:				
baseType	ErrorType	O	1	Reference to a RoLo error type that is the base type of this RoLo error type. This attribute is used to represent hierarchical relationships among RoLo error types.

Table 7.34 - ErrorTypeOperation class

Description: Denotes transformation of RoLo error into a different RoLo error type.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes:				
source	ErrorType	M	1	Source RoLo error type
target	ErrorType	M	1	Target RoLo error type

7.3.4.2 RoLo Error

RoLo errors are objects for holding error information in different representations. Here we define some frequently used forms. Users may extend these classes to implement their own RoLo error containers, accompanied with appropriate RoLo error type definitions.
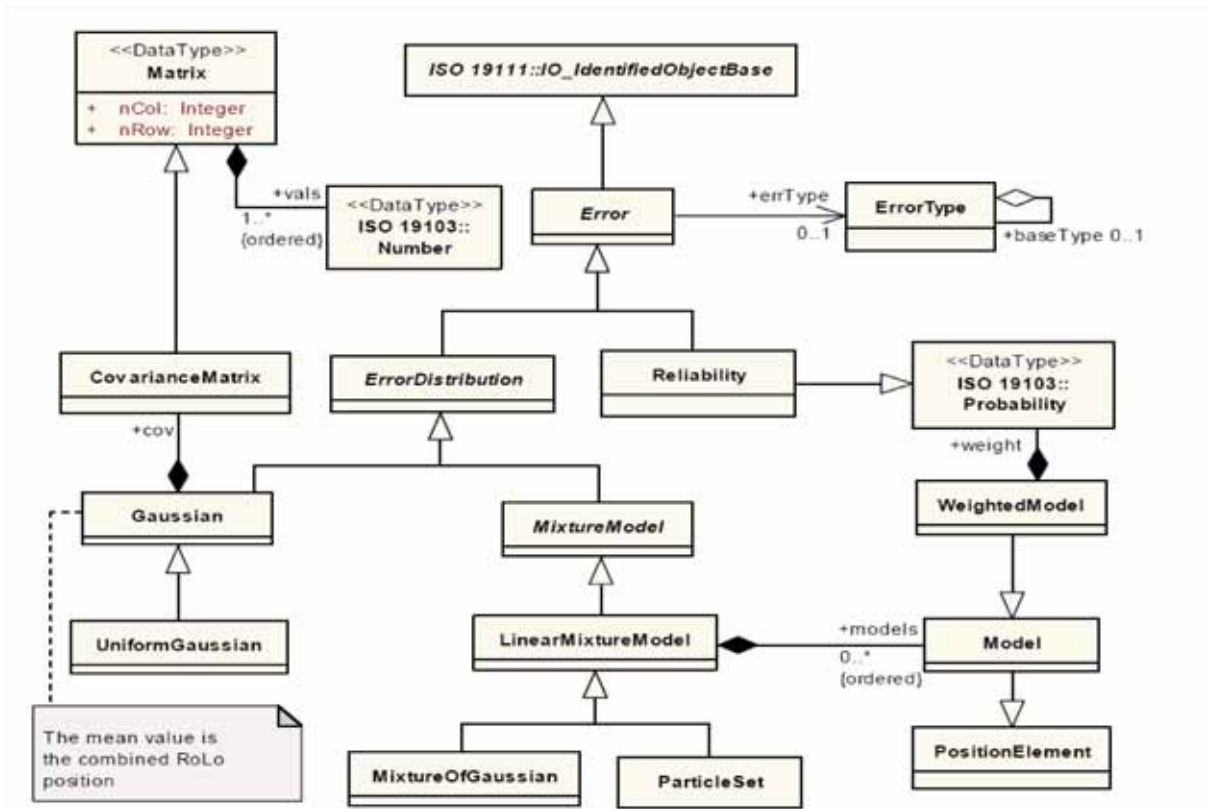
Figure 7.10 - RoLo Error

Table 7.35 - Error class

Description: Base abstract class for holding error information				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes:				
errType	ErrorType	O	1	Reference to the RoLo error type indicating how this error information is represented.

Table 7.36 - Reliabilityclass

Description: Reliability value. The derived attribute 'errType' shall be ET_Reliability.
Derived From: Error, Probability [ISO19103]

Table 7.37 - ErrorDistributionclass

Description: Base abstract class for error information reprsented by a probability distribution.
Derived From: Error

Table 7.38 - Matrix class

Description:	N-dimensional matrix				
Derived From:	none				
Attributes:					
nRow	Integer	M	1	Number of matrix rows. The value of attribute 'nRow' should be a positive integer.	
nCol	Integer	M	1	Number of matrix columns. The value of attribute 'nCol' should be a positive integer.	
vals	Number [ISO19103]	M	N ord	Value elements of the matrix.	

Table 7.39 - CovarianceMatrix class

Description: An n-dimensional matrix describing covariance
Derived From: Matrix
Note – This shall represent a square matrix where nRow = nCol.

Table 7.40 - Gaussian class

Description: Error represented by an n-dimensional normal distribution. The mean value is denoted by the accompanying RoLo position. The derived attribute 'errType' shall be ET_Gaussian.				
Derived From: ErrorDistribution				
Attributes:				
cov	CovarianceMatrix	M	1	Indicates the covariance for the normal distribution.

Table 7.41 - UniformGaussian class

Description: Error represented by a uniform normal distribution.
Derived From: Gaussian
Note – Dimensions of the cov attribute derived from class Gaussian shall all be equal to 1. That is, nRow = nCol = 1.

Table 7.42 - ParticleSet class

Description: Error represented by a set of particles. As for the 'models' attribute derived from LinearMixtureModel class, the 'postElem' attribute shall either have no 'err' attribute or have a RoLo error like an impulse response (such as a Gaussian distribution with zero standard deviation). Normally, this is used for representing distributions by Monte Carlo approximation, where distributions are approximated by a finite number of random samplings. The derived attribute 'errType' shall be ET_ParticleSet.
Derived From: LinearMixtureModel

Table 7.43 - MixtureModel class

Description: Abstract base class for representing an error distribution by means of mixture of probability distributions.
Derived From: ErrorDistribution

Table 7.44 - Model class

Description: A distribution without a weight Recall that a PositionElement object can be interpreted to represent a probability distribution. Its 'pos' attribute is treated as the expected coordinate value and its 'err' attribute as the shape of distribution.
Derived From: PositionElement

Table 7.45 - WeightedModel class

Description: A distribution with a weight.				
Derived From: Model				
Attributes:				
weight	Probability [ISO19103]	M	1	Weight of this distribution.

Table 7.46 - LinearMixtureModel class

Description: Base abstract class for representing distributions represented by linear mixture of probability distributions. The derived attribute 'errType' shall be ET_LinearMixtureModel.				
Derived From: MixtureModel				
Attributes:				
models	Model	M	N ord	List of models to be combined. When the models are non-weighted, they shall be treated to have equal weights.

Table 7.47 - MixtureOfGaussian class

Description: A distribution represented by a linear mixture of Gaussian distributions. The derived attribute 'errType' shall be ET_MixtureOfGaussian. The models attribute derived from LinearMixtureModel shall have a 'posElem' attribute whose 'err' attribute is restricted to be an instance of Gaussian class.
Derived From: LinearMixtureModel

7.3.5   Robotic Localization Architecture

The *Robotic Localization* (*RoLo) Architecture* defined here is a unified framework for organizing and representing complex data set required in robotic localization. Similar to the relation between GIS location data and coordinate reference system, two sets of structures are defined here:

1.   Classes for holding the localization results (Data, Element, and Position).

2.   Classes for describing the structure or the meaning of localization results (DataSpecification, ElementSpecification).

These two sets of classes are in relation similar to that between GIS position data and coordinate reference systems: the latter describes the structure and meaning of the former. The RoLo element and RoLo element specification pair binds the main localization data element to error information. The RoLo data and RoLo data specification pair defines the structure and relation among a set of RoLo elements that forms a complete robotic localization result.

Normally, error information is combined with one main localization element. However, in certain cases, there is a need to hold an integrated error among multiple location data. For example, in a typical Kalman filter usage, multiple main location information such as spatial position and velocity are used to form a state vector. When the elements of the state vector are not independent, which is the usual case, the corresponding error, the covariance matrix, is related to multiple main elements. In such case, the ErrorElementSpecification (derived from ElementSpecification class) specifies which main information slot the error is related to, and the actual error data is contained by the ErrorElement class (derived from Element class) instances. Figure 7.11 shows a sample data structure and corresponding object diagram.
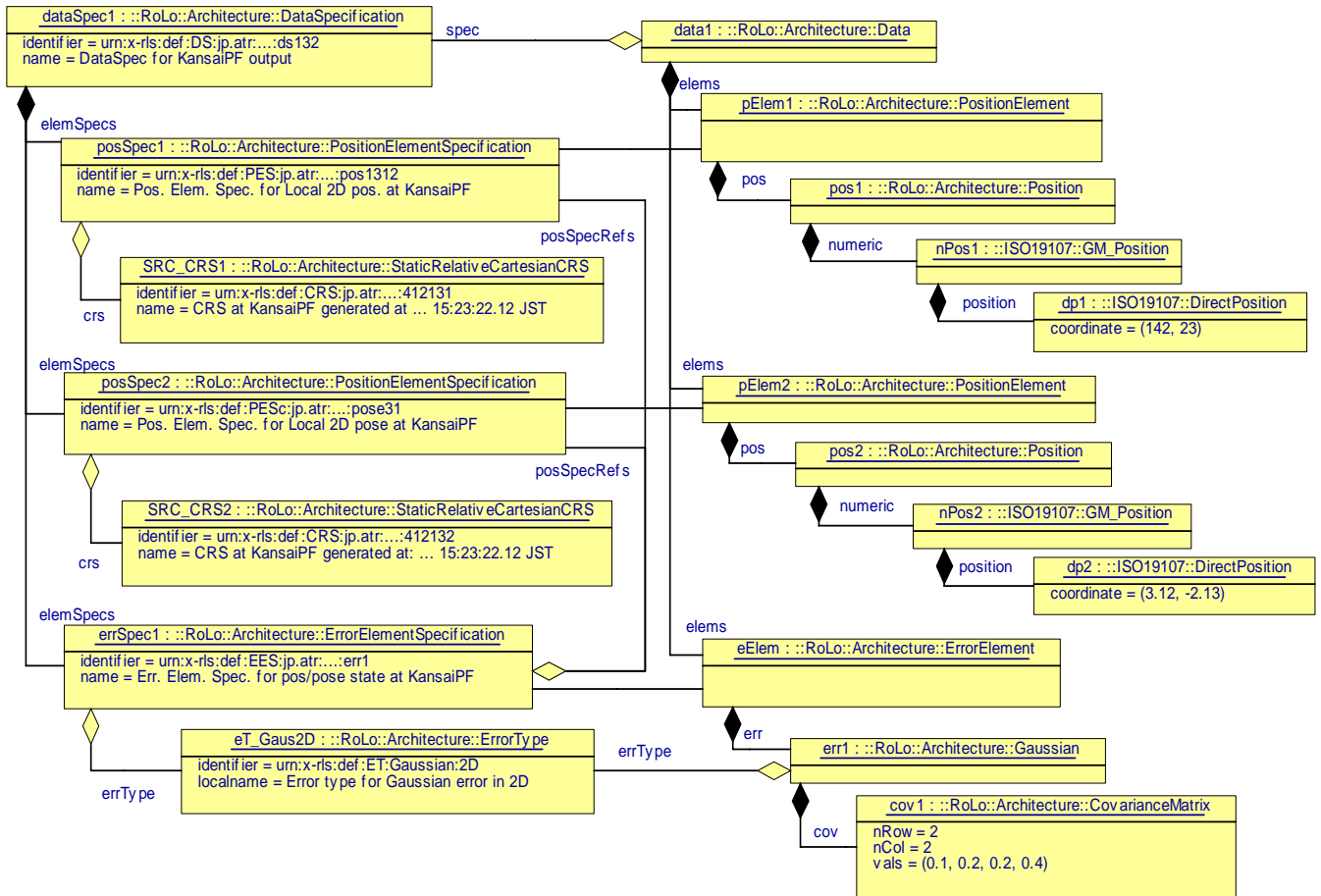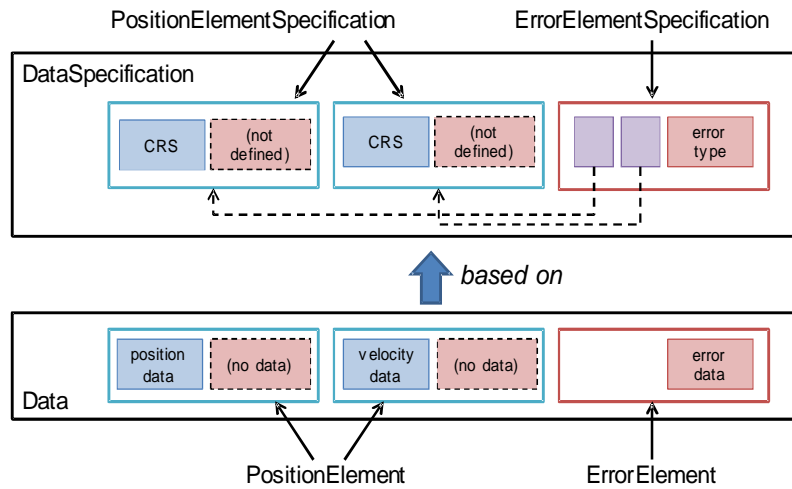
Figure 7.11 - Representation of error information related to multiple localization data
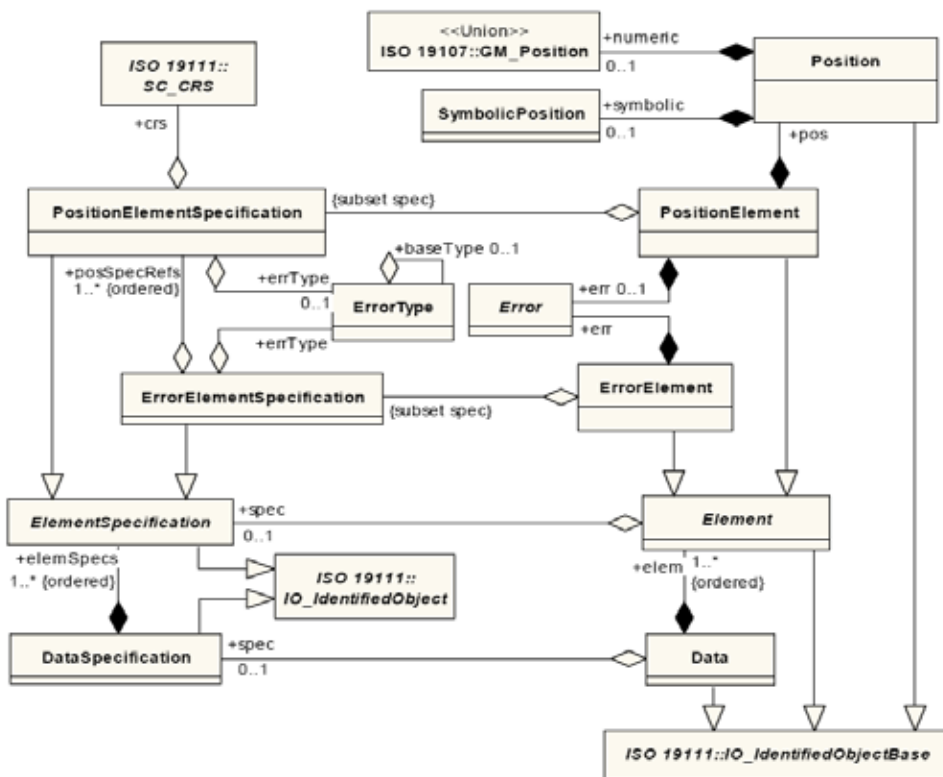
Figure 7.12 - RoLo Architecture

Table 7.48 - Position class

Description: Data container for localization results without error information. This is formed as a union of SymbolicPosition class and GM_Position [ISO19107] class. The former is a container for symbolic symbols such as identity information, and the latter contains numerical data such as spatial coordinate values.				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes:				
symbolic	SymbolicPosition	C	1	Symbolic data container
numeric	GM_Position [ISO19107]	C	1	Numeric data container
Condition:	One and only one of the choices shall be chosen.			

Table 7.49 - ElementSpecification class

Description: Base abstract class for holding structural definition for RoLo elements. Instances of this class contain meta-level information on what kind of data each RoLo element holds.
Derived From: IO_IdentifiedObject [ISO19111]

Table 7.50 - Position class

Description: Data container for localization results without error information. This is formed as a union of SymbolicPosition class and GM_Position [ISO19107] class. The former is a container for symbolic symbols such as identity information, and the latter contains numerical data such as spatial coordinate values.				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes:				
symbolic	SymbolicPosition	C	1	Symbolic data container
numeric	GM_Position [ISO19107]	C	1	Numeric data container
Condition: One and only one of the choices shall be chosen.				

Table 7.51 - PositionElementSpecification class

Description: Specification holder for RoLo position elements				
Derived From: ElementSpecification				
Attributes:				
crs	SC_CRS [ISO19111]	M	1	Reference to a coordinate reference system that the 'pos' attribute in RoLo position element is based on.
errType	ErrorType	O	1	Reference to a RoLo error type. Specifies the type of 'err' attribute in RoLo position elements. If this attribute is omitted, RoLo position elements related with this instance shall not contain error information.

Table 7.52 - ErrorElementSpecification class

Description: Definition holder for RoLo error elements				
Derived From: ElementSpecification				
Attributes:				
posSpecRefs	PositionElement Specification	M	Nord	An ordered list of references to RoLo position element specifications showing which positional data the RoLo error contained in the RoLo error element is related to. The referred RoLo position element specifications shall be contained in the same RoLo data specification as this class instance.
errType	ErrorType	M	1	Reference to a RoLo error type. Specifies the type of 'err' attribute in RoLo error elements.

Table 7.53 - Element class

Description: Base abstract class for RoLo elements that holds the binding between the main positional data and the RoLo error.				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes:				
spec	ElementSpecification	O	1	Reference to RoLo element specification that this element is based on.

Table 7.54 - PositionElement class

Description: Data container of each localization result by combining the main positional data and the accompanying RoLo error.				
Derived From: Element				
Attributes:				
pos	Position	M	1	The main information
err	Error	O	1	RoLo error information related to the 'pos' attribute of the same instance. If the RoLo position element specification referred related with this instance does not hold an 'errType' attribute, this attribute shall be omitted.
Note – Values for the attribute 'spec' that are derived from parent class shall be limited to instances of PositionElementSpecification or its inherited classes.				

Table 7.55 - ErrorElement class

Description: Data container of error information that is related to multiple positional data in the same RoLo data. RoLo position elements related with this error information are specified in the referenced RoLo error element specification.				
Derived From: Element				
Attributes:				
err	Error	M	1	RoLo error bound with the specified RoLo position elements.
Note – Values for the attribute 'spec' that are derived from parent class shall be limited to instances of ErrorElementSpecification or its inherited classes.				

Table 7.56 - DataSpecification class

Description: Specification holder for RoLo data				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes:				
elemSpecs	ElementSpecification	M	N ord	Ordered list of RoLo element specifications that defines the structure of localization result.

Table 7.57 - Data class

Description: Data container for the robotic localization result				
Derived From: IO_IdentifiedObjectBase [ISO19111]				
Attributes:				
spec	DataSpecification	O	1	Reference to the corresponding RoLo data specification
elems	Element	M	N ord	An ordered list of RoLo elements. Numbers, orders, and types of the RoLo elements shall match that of the corresponding RoLo data specification.

7.3.5.1 Don't-Care

To handle generic data specifications, specifications may include "don't care" values in their definition. For example, you may want to build a people tracking service that accepts outputs from another RoLo module bound with a camera sensor and performs some calculation. In such case, the coordinate system of the camera sensor output may be fixed but the coordinate reference system and the datum associated with each camera module may differ, depending on the location where the camera is installed. Building such module is impossible in the normal RoLo framework, as each RoLo stream needed to clearly specify a set of RoLo data specifications it can accept; you need to specify an infinite list of RoLo data specifications on the input stream ability description.

That's where don't-cares are used. In such cases, you specify a RoLo data specification for the tracking module's input stream ability using a coordinate reference system that uses a don't-care datum (NULLDatum class). This way you can specify only the specification parts you (the module) is interested, and leave the other parts free. Such is quite a common usage, and so the use of don't-cares will increase the flexibility and usability of the RoLo service. However, this use of don't-care elements may require notice as it may result in high computation cost or ambiguous, useless specifications that break the idea of having specifications for data. Thus, we need some rule to avoid misleading usages. The following describes the rules that shall be followed on using don't-cares:

- When multiple type specifications are associated with a data instance, the specifications shall be consistent with each other.

- Every data instance shall have a complete unified type specification.

- A type specification may include don't-cares for the following attributes:

    - 'elemSpec' in RoLo data specification

    - 'crs' in RoLo position element specification

    - 'errType' in RoLo position element specification or RoLo error element specification

    - 'cs' in SC_CRS [ISO19111]

shows the classes and the objects used to indicate don't-care:

    - 'datum' in SC_CRS [ISO19111]

The last rule means that any RoLo stream that is associated with a complete RoLo data specification may skip checking explicit specification of each RoLo data or its subcomponents passed through itself. Thus, modules equipped with low computation power can avoid unnecessary processing by specifying explicit data specifications as their RoLo input stream ability.
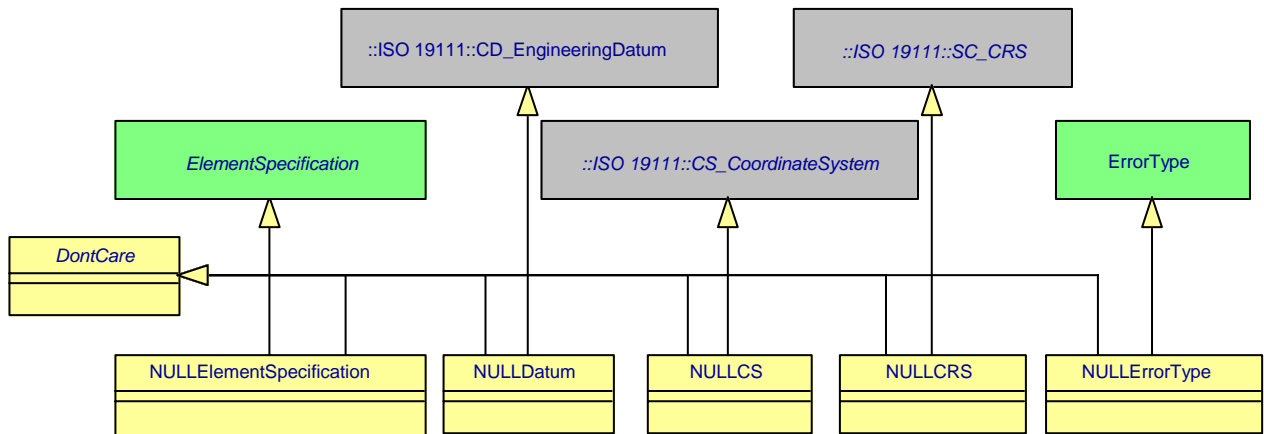
(a) Don't care classes
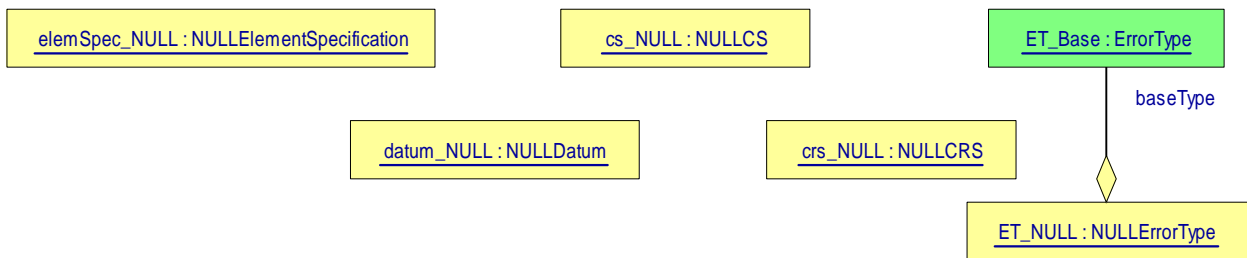
(b) Don't care objects

Figure 7.13 - Don't-care classes and objects

Table 7.58 - Don'tCare class

Description: Base abstract class for don't-care classes	
Derived From: (none)	

Table 7.59 - NULLCS class

Description: Don't-care indicator. Used for indicating that this coordinate system shall be ignored.
Derived From: DontCare, CS_CoordinateSystem [ISO19111]

Table 7.60 - NULLCRS class

Description: Don't-care indicator. Used for indicating that this coordinate reference system shall be ignored.
Derived From: DontCare, SC_CRS [ISO19111]

Table 7.61 - NULLDatum class

Description: Don't-care indicator. Used for indicating that this datum shall be ignored.
Derived From: DontCare, CD_EngineeringDatum [ISO19111]

Table 7.62 - NULLErrorType class

Description: Don't-care indicator. Used for indicating that this RoLo error type shall be ignored.
Derived From: DontCare, ErrorType

Table 7.63 - NULLElementSpecification class

Description: Don't-care indicator. Used for indicating that this slot in RoLo element specification shall be ignored.
Derived From: DontCare, ElementSpecification

7.3.5.2 RoLo Data Operation
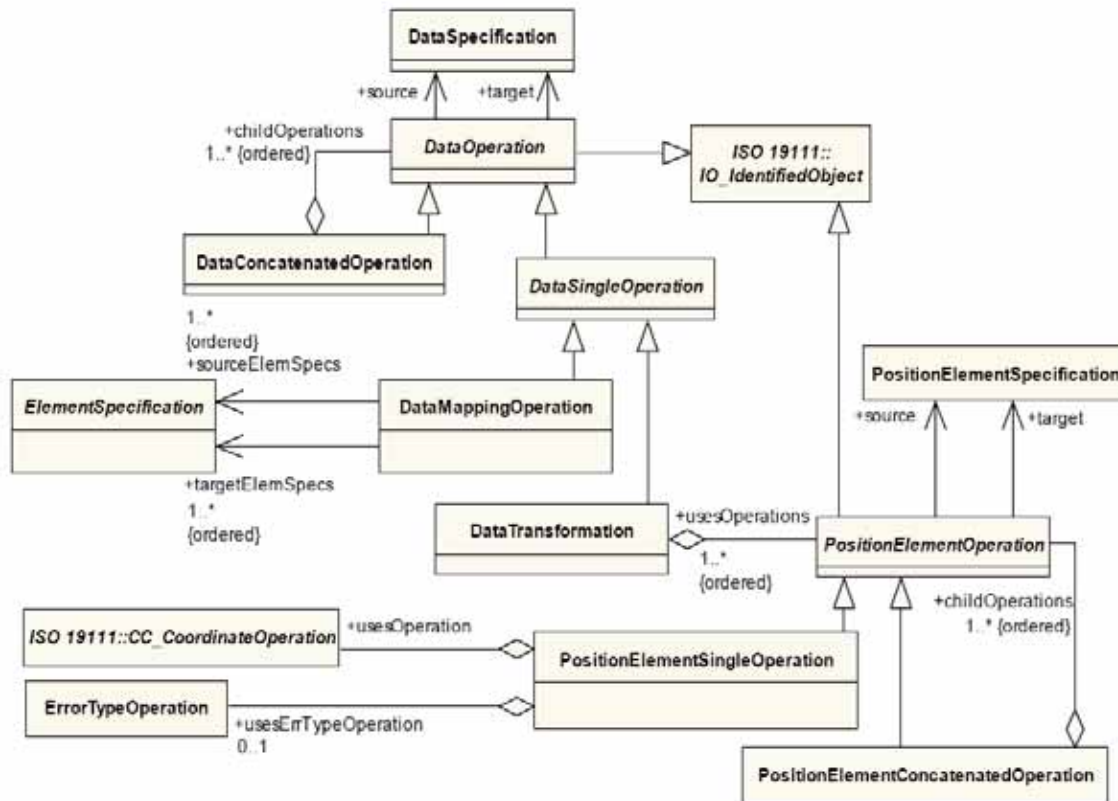
Figure 7.14 - RoLo Data Operation

Table 7.64 - PositionElementOperation class

Description: Base abstract class for representing operations for transforming data between different RoLo position elements. RoLo position elements are basically composed by RoLo position and RoLo error. As the value of RoLo error is also based on the coordinate reference system where the combined RoLo position is based on, both the main information and the error information shall be transformed at once.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes:				
source	PositionElementSpecification	M	1	Source RoLo position element specification
target	PositionElementSpecification	M	1	Target RoLo position element specification

Table 7.65 - PositionElementConcatenatedOperation class

Description: Concatenation of multiple PositionElementOperation instances.				
Derived From: PositionElementOperation				
Attributes:				
childOperations	PositionElementOperation	M	N ord	Ordered list of PositionElementOperation to be applied. Target RoLo position element specification and source RoLo position element specification for succeeding operations shall match.

Table 7.66 - PositionElementSingleOperation class

Description: Definition of an operation for transforming or converting data between different RoLo position element specifications. The main information is processed by the CC_CoordinateOperation [ISO19111], and the error information should also be transformed.				
Derived From: PositionElementOperation				
Attributes:				
usesOperation	CC_CoordinateOperation [ISO19111]	M	1	Operation to be used for transforming the main localization data. This operation may also be utilized to transform the accompanying RoLo error.
usesErrTypeOperation	ErrorTypeOperation	O	1	Operation to be used for converting the type of the RoLo error part. If no error type conversion is necessary, this part may be omitted.

Table 7.67 - DataOperation class

Description: Base abstract class for representing operations for transforming data between different RoLo data specifications. The main purpose of this operation is to transform or to convert RoLo data that contains RoLo error element. RoLo data that contains RoLo error element needs to know about how other elements within the same RoLo data specification are operated. Instances of this class perform.				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes:				
source	DataSpecification	M	1	Reference to the originate RoLo data specification
target	DataSpecification	M	1	Reference to the target RoLo data specification

Table 7.68 - DataConcatenatedOperation class

Description: Concatenation of multiple RoLo data operations				
Derived From: DataOperation				
Attributes:				
childOperations	DataOperation	M	N ord	Ordered list of RoLo data operation to be applied. Target RoLo data specification and source RoLo data specification for succeeding operations shall match.

Table 7.69 - DataSingleOperation class

Description: Abstract class for representing an operation for transforming data between different RoLo data specifications.
Derived From: DataOperation

Table 7.70 - DataTransformation class

Description: Definition of an operation for transforming data between different RoLo data specifications.				
Derived From: DataSingleOperation				
Attributes:				
usesOperations	PositionElementOperation	M	N ord	Operations used for each of the RoLo position element specification in the RoLo data specification. The number of RoLo position element specifications in this RoLo data specification and that of 'usesOperation' attribute shall match. The operation defined here is applied to each of the RoLo position elements in the order the corresponding RoLo position element specifications are defined.

Table 7.71 - DataMappingOperation class

Description: Definition of an operation for transforming data between different RoLo data specifications that simply maps elements in the source RoLo data specification to elements in the target RoLo data specification. Only the structures of the RoLo elements are altered, and the data content itself are not changed. With RoLo error elements, the reference to the RoLo position elements shall be modified appropriately. The two attributes contained are lists of references to RoLo element specifications in source and target RoLo data specifications that defines how the mapping is to be performed.				
Derived From: DataSingleOperation				
Attributes:				
sourceElemSpecs	ElementSpecification	M	N ord	Ordered list of RoLo element specification references within the source RoLo data specification that is to be mapped to the RoLo element specification in the target RoLo data specification represented by the 'targetElemSpecs' attribute value at the same position. The numbers of 'sourceElemSpecs' attribute shall match that of 'targetElemSpec' attribute.
targetElemSpecs	ElementSpecification	M	N ord	Ordered list of RoLo element specification references within the target RoLo data specification.

7.4 DataFormat Package

When exchanging information among modules, knowledge on data structures is not enough. We need to specify the actual data representation format exchanged among modules.
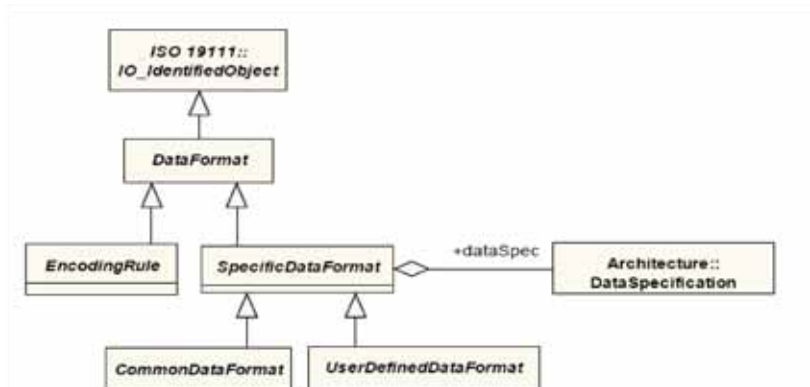
Figure 7.15 - RoLo Data Format

When exchanging information among modules, knowledge on data structures is not enough. We need to specify the actual data representation format exchanged among modules.

Table 7.72 - DataFormat class

Description: Base abstract class for data format definitions	
Derived From: IO_IdentifiedObject [ISO19111]	

Table 7.73 - EncodingRule class

Description: Base abstract class for encoding rules. Encoding rule denotes some systematic mean that can determine the data format from corresponding data structure (i.e., RoLo data specification). Packed Encoding Rule [PER] is an example of encoding rule. This is a reserved class for future extension.
Derived From: DataFormat

Table 7.74 - SpecificDataFormat class

Description: Abstract class for data formats where format description is tightly coupled with data structure. This is in contrast with the EncodingRule class, where data formatting rules are independent to data structure definitions.				
Derived From: DataFormat				
Attributes:				
dataSpec	DataSpecification (RoLo::Architecture)	M	1	Specifies a RoLo data specification that this data format can handle.

Table 7.75 - UserDefinedDataFormat class

Description: Abstract class for user-defined, non-common data formats	
Derived From: SpecificDataFormat	

Table 7.76 - CommonDataFormat class

Description: Abstract class for denoting Common Data Formats	
Derived From: SpecificDataFormat	

7.4.1   Common Data Format

This specification allows a wide range of data formats for keeping compatibility to widely used data formats. This specification, however, defines three common data formats each with two different RoLo data specifications, representing location information in order to provide interoperability between modules that have a lack of computing resources. Every module in RoLo service shall support at least one of these common data formats in order to transmit location information to enhance inter-module connectability as much as possible.

In this specification, depending on the coordinate systems to refer the position and the methods to specify the orientation, the common data format is represented by one of the six types, Type I-1, I-2, II-1, II-2, III-1, and III-2 as follows.

7.4.1.1   Type I-1

Table 7.77 - Common data format type I-1 (Cartesian Coordinate System, PT_Euler_xyz Pose Representation)

Parameter	Format of value	Value type	Unit
Position	[x, y, z]	Real, Real, Real	meter, meter, meter
Orientation	[α, β, γ]	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

7.4.1.2   Type I-2

Table 7.78 - Common data format type I-2 (Cartesian Coordinate System, PT_Euler_zY1X2 Pose Representation)

Parameter	Format of value	Value type	Unit
Position	[x, y, z]	Real, Real, Real	meter, meter, meter
Orientation	[yaw α, pitch β, roll γ]	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

7.4.1.3 Type II-1

Table 7.79 - Common data format type II-1 (Spherical Coordinate System, PT_Euler Angle_xyz Pose Representation)

Parameter	Format of value	Value type	Unit
Position	$[r, \theta, \varphi]$	Real, Real, Real	meter, radian, radian
Orientation	$[\alpha, \beta, \gamma]$	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

7.4.1.4 Type II-2

Table 7.80 - Common data format type II-2 (Spherical Coordinate System, PT_Euler Angle zY1X2 Pose Representation)

Parameter	Format of value	Value type	Unit
Position	$[r, \theta, \varphi]$	Real, Real, Real	meter, radian, radian
Orientation	[yaw α, pitch β, roll γ]	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

7.4.1.5 Type III-1

Table 7.81 - Common data format type III-1 (Geodetic Coordinate System, PT_Euler_xyz Pose Representation)

Parameter	Format of value	Value type	Unit
Position	[latitude φ, longitude λ, height h]	Real, Real, Real	degree, degree, meter
Orientation	$[\alpha, \beta, \gamma]$	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

7.4.1.6 Type III-2

Table 7.82 - Common data format type III-2 (Geodetic Coordinate System, PT_Euler_zY1X2 Pose Representation)

Parameter	Format of value	Value type	Unit
Position	[latitude φ, longitude λ, height h]	Real, Real, Real	degree, degree, meter
Orientation	[yaw α, pitch β, roll γ]	Real, Real, Real	radian, radian, radian
Timestamp	POSIX time	Integer, Integer	second, nanosecond
ID	--	Integer	--

Each type of the common data formats includes four parameters, as follows:

1. ***Position*** – specifies the coordinate value in a Cartesian coordinate system for Type I-1 and I-2, in a spherical coordinate system for Type II-1 and II-2, and in a geodetic coordinate system for Type III-1 and III-2. (See Figure 7.16 and its explanation for details).

2. ***Orientation*** – specifies sequential three rotations by each axis in a right-handed 3-dimensional Cartesian coordinate system defined by the PT_Euler_xyz pose representation for Type I-1, II-1, and III-1 and by the PT_Euler zY1X2 pose representation for I-2, II-2, and III-2.

3. ***Timestamp*** – specifies time at occurring measurement for current position and orientation. It is compatible to POSIX time that is the time elapsed since midnight Coordinated Universal Time (UTC) of January 1, 1970. A timestamp consists of two integers of elapsed seconds and nanoseconds that is compatible to standard UNIX C time_t data structure.

4. ***ID*** – specifies the identifier of current location information for robots and related entities.

The coordinate values of position information in the common data format in Table 7.71 to Table 7.76 are defined respectively by three different coordinate systems: Cartesian coordinate, spherical coordinate, and geodetic coordinate system as shown in Figure 7.16.
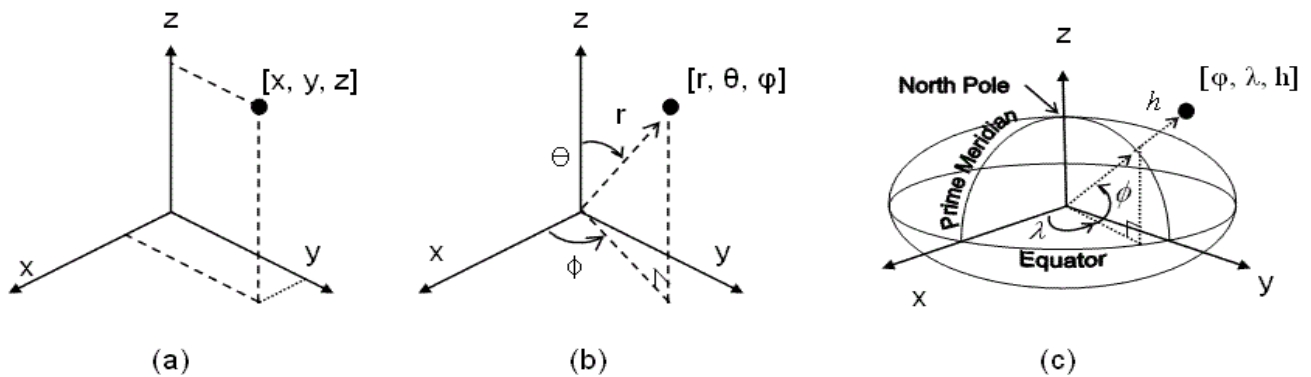
Figure 7.16 - Definition of a position and reference coordinate systems used in the common data format: (a) Cartesian coordinate system for Type I-1 and I-2, (b) spherical coordinate system for Type II-1 and II-2, (c) geodetic coordinate system for Type III-1 and III-2

7.5   Filter Condition Package

When a location service is operated on a large scale and handles a large number of location information, it is useful that the service has a filtering functionality by which it limits outgoing RoLo data by a given condition. Without this functionality, service providers and receivers are required to have large capacities of output/input to process the whole data from large scaled systems. Suppose, as an example, that we implement a sensor system at a shopping center that detects thousands of guests at once and provides localization service for robots. In such case, it is not reasonable for each robot to receive localization data about the whole guests every time. Instead each robot is generally interested in specific guests identified by certain features, area, and/or time period.

The "Filter Condition" specified below is aimed to provide the functionality for localization services to specify a condition for filtering data sent to service receivers.

7.5.1   Filter Condition in RoLo stream

A RoLo output stream may have the functionality to filter localization results by a certain condition. We call this condition as "filter condition." When a filter condition is specified, each localization result is tested by the condition and passed to the output stream only when it satisfies the condition.

If no condition is given, or if the stream has no such functionality, the "True" condition is used as the default condition, in which all localization results are passed to the output stream.

To handle the filter condition functionality, ability descriptor for RoLo streams shall additionally have the following parameters.

Table 7.83 - Filter Condition parameter for RoLo streams

filterCondition	Parameter<::ISO19143::NonIdOperator> (RoLo::Interface)	O	1	Filter condition to be used for output data. Default value is "True."

Users can set and get the content of this parameter through the 'setParameterValueSet' and 'getParameterValueSet' methods toward the stream or the service. When filter condition is not supported by the stream, UNSUPPORTED_PARAMETER will be returned.

7.5.1.1 Data Format of Filter Condition

To specify a filter condition, we follow the ISO 19143 specification [ISO/DIS19143] that is defined for ISO 19142 [ISO/DIS19142]. ISO 19143 specifies XML encoding and UML class charts of filter conditions and their operators.

While the UML charts provides general concepts of data format of the filter condition, it is generally useful and flexible enough to use the XML encoding for the localization service. (see Examples.)

7.6   Interface Package

Several types of modules are commonly used in robotic localization services in general. The simplest form of module is that which receives data from sensors, calculates location, and outputs the results. However, this type of interface strongly depends on sensor interfaces or sensor output formats. Strong dependency on specific products or vendors is not suitable for standardization. Moreover, when a location is calculated, many kinds of resources such as map data, specific to each sensing system, are required. It is impractical to include each of these resources into the standard specification. Thus, we decided to embed and hide the individual device or localization algorithm details inside the module structure (Figure 7.17).
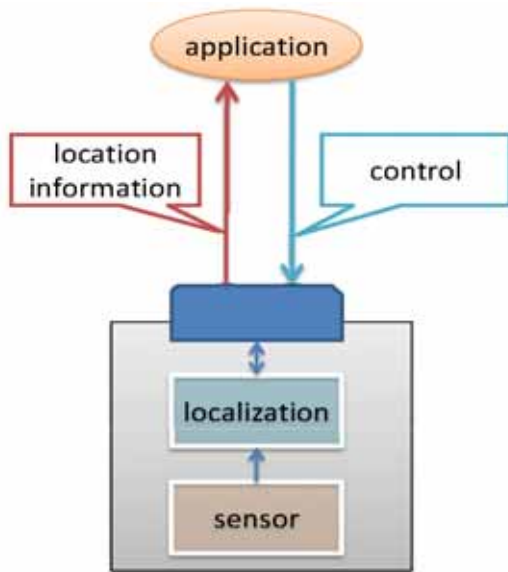
Figure 7.17 - Basic robotic localization module

On the other hand, if we focus on functionality required to localization modules, we can classify them into roughly three classes (Figure 7.18):

A)   Calculate localization results based on sensor outputs (measurement)

B)   Aggregate or integrate multiple localization results (aggregation)

C)   Transform localization results into different coordinate reference systems (transformation)
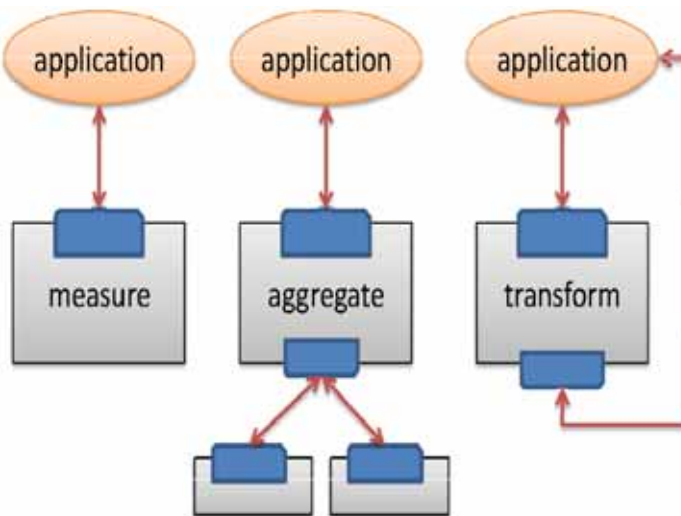
Figure 7.18 - Structures of robotic localization module with different functionalities

These functionalities differ in their internal algorithms or the number of input / output streams. However, in all of these, the main data to be exchanged is localization results. As we are focusing on the interface of RLS modules, and not on their functionalities, we decided to abstract these different types of modules into a single form of module. This abstract module holds n (>=0) input streams and a uniform output stream. By abstracting various types of modules and assuming a uniform interface, complex module compositions such as hierarchical or recursive module connections can be easily realized (Figure 7.19).
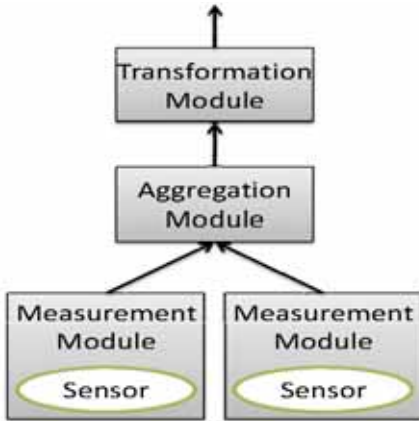
Figure 7.19 - Example of a cascading module connection

A RoLo service (implemented as a Service class) may have one or more RoLo output streams (OutStream class) and zero or more RoLo input streams (InStream class). Typically, the number of RoLo inputs a service owns is predetermined and the number of RoLo outputs a service owns changes dynamically based on requests from service users. This is similar to typical server systems such as database or Web servers where the number of established output connection increases as requests arrive until it reaches a predefined maximum number.

If each module can represent what or how it can perform, or provide information on available configurable parameters, a large amount of development efforts can be reduced. Thus, each service or stream is modeled to own an *ability* description (Ability class) that contains a set of *attributes* (Attribute class) and *parameters* (Parameter class). Attributes show some static nature of a module and parameters indicate its configurable parameters. For example, an ability description for a service (ServiceAbility class) includes an attribute describing expected value of latency. And an ability description for a stream (StreamAbility class) includes parameters denoted by lists of DataSpecification and DataFormat objects that shows what type of data structure or data format a stream can handle, respectively. Attributes or parameters specific to each implementation, such as vendor-specific parameters, can be described by extending the respective classes. As such, attributes may be used to describe fixed nature (catalog specs) of modules, while parameters define configurable settings for modules. Note that some parameters may not be configurable on some implementations. For example, if a module implementation can output data only by a single data format, the aforementioned parameter for DataFormat may show only a single candidate, and be marked as non-configurable (Parameter.isConfigurable = false).

Often, parameters are defined over some limited value domains. As in the example given above, data specifications or data formats that a stream is able to pass data are likely to be limited to sets with a small number of choices. Or some parameters, such as output frequency, may be restricted under a limited range of values. The attribute 'domain' in ParameterOverDomain class is aimed to denote these limitations. As the value domain required may take variations of forms such as finite set or interval (or range), the ParameterOverDomain class is defined as a template class that allows a type argument for indicating what sort of value domain shall be specified.

By defining the "meaning" of attributes and parameters, the ambiguity in functional definition or parameters can be eliminated which can be expected to increase developing efficiency. For example, what does the value 0.23 given as an 'expectedError' attribute for a RoLo Service mean? These ambiguities can often be seen in sensor products such as GPS receivers, making it difficult to design a reusable system applicable to devices or modules from different vendors. The AttributeDefinition class is aimed to clarify the meaning of attributes and parameters. Although this is out of scope for this specification, by providing a repository of AttributeDefinition objects that can be referred on demand, RoLo service users and developers can always make sure what each ability description means or on which unit they are defined.

Moreover, advanced features can be implemented such as verification of inter-module connection, automatic search of specific modules or semi-automatic parameter negotiation between modules. In cases where sensors or robots distributed in the environment cooperate with each other, namely the Network Robot environment, it becomes essential to register each module's capabilities in repositories and make them searchable.
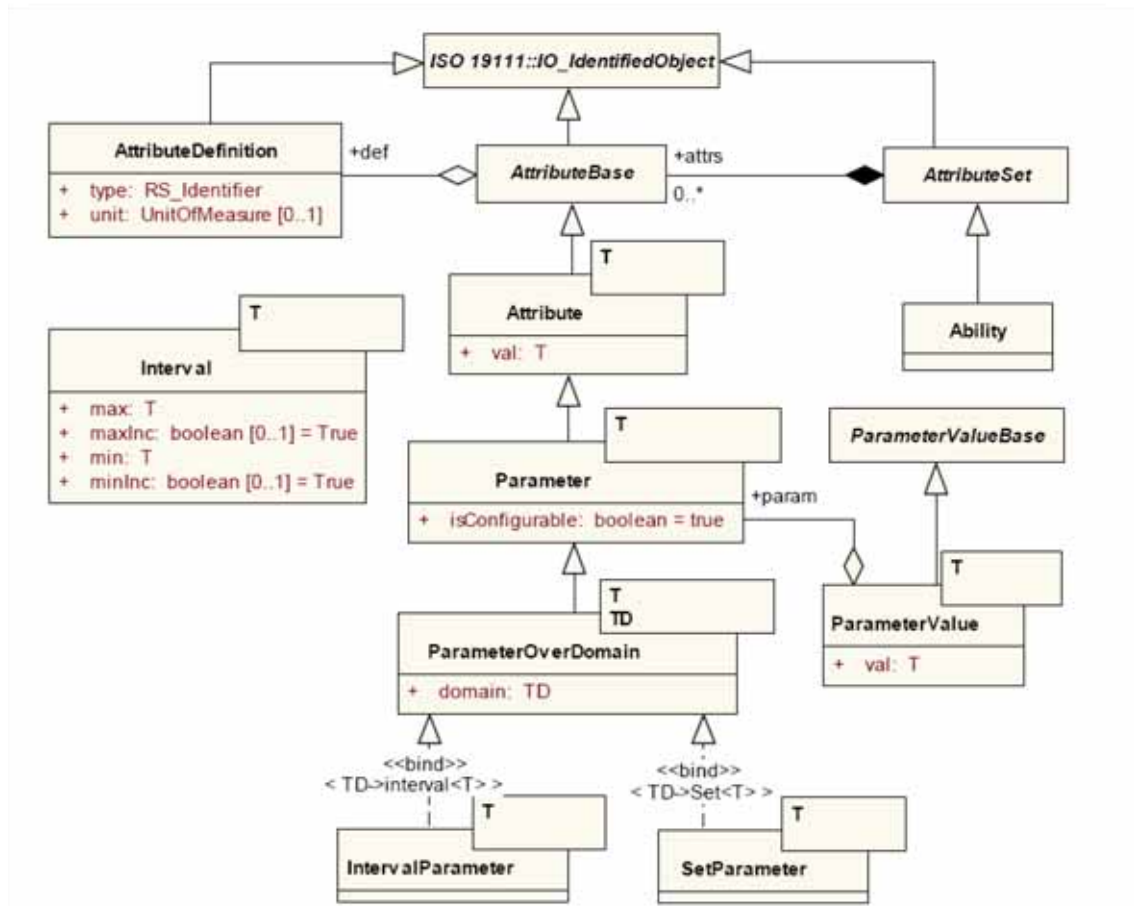
Figure 7.20 - RoLo Ability

Table 7.84 - AttributeDefinition class

Description:	Definition of a single attribute			
Derived From:	IO_IdentifiedObject [ISO19111]			
Attributes:				
type	RS_Identifier [ISO19115]	M	1	Type descriptor for this attribute.
unit	UnitOfMeasure [ISO19103]	O	1	Unit of the target attribute. If no unit is required, this may be omitted.

Table 7.85 - AttributeBase class

Description:	Base abstract class for different types of Attribute classes			
Derived From:	IO_IdentifiedObject [ISO19111]			
Attributes:				
def	AttributeDefinition	M	1	Reference to an AttributeDefinition object indicating definition for this attribute.

Table 7.86 - Attribute class

Description:	Represents a single attribute. This is a template class with type argument T that denotes the type of attribute value. The type argument T shall be consistent with the value of 'type' attribute in AttributeDefinition object referred by the 'def' attribute derived from AttributeBase class.			
Derived From:	Attribute			
Attributes:				
val	T	M	1	Value of this attribute

Table 7.87 - Parameter class

Description:	Represents a single parameter. A parameter is an attribute that may be configurable. This is a template class with type arguments T that denotes the type of parameter value.			
Derived From:	Attribute <T>			
Attributes:				
isConfigurable	Boolean	O	1	Flag to show whether this parameter is configurable or not. If omitted, assumed to True. When this value is set to False, this parameter is not configurable.

Table 7.88 - ParameterOverDomain class

Description: Represents a parameter whose value domain is defined. This is a template class with type arguments T and TD, where T denotes the type of parameter value and TD denotes the type to show domain of the parameter value.				
Derived From: Parameter <T>				
Attributes:				
domain	TD	M	1	Domain of parameter value

Table 7.89 - Interval class

Description: Class for indicating an interval. Note that an interval is sometimes referred to as a 'range.'				
Derived From: (none)				
Attributes:				
min	T	M	1	Minimum value of interval
max	T	M	1	Maximum value of interval
minInc	Boolean	O	1	Flag to show whether the minimum value is included in the range. Default is True.
maxInc	Boolean	O	1	Flag to show whether the maximum value is included in the range. Default is True.

Table 7.90 - IntervalParameter class

Description: A parameter whose value domain is defined as an interval. This is a template class with type argument T. The type argument TD from ParameterOverDomain is deduced to be class Interval <T>.
Derived From: ParameterOverDomain<T, Interval>

Table 7.91 - SetIParameter class

Description: A parameter whose value domain is defined as a set of values. This is a template class with type argument T. The type argument TD from class ParameterOverDomain is deduced to be a set.
Derived From: ParameterOverDomain<T, Set<T>>

Table 7.92 - ParameterValueBase class

Description: Base abstract class for different types of ParameterValue class
Derived From: (none)

Table 7.93 - ParameterValue class

Description: A Class that represents values for parameters. This is a template class with type argument T that denotes the type of the parameter value. This class is typically used to set/get parameter values in the Interface::Stream instances through setParameterValues() and getParameterValues() methods in Interface::InterfaceBase class. The actual values are stored in the 'val' attribute of Interface::Attribute class.				
Derived From: AttributeBase				
Attributes:				
val	T	M	1	Value of the parameter
param	Parameter	M	1	Reference to a Parameter object this parameter value is for. The template argument T for this class shall match the template argument of the referred Parameter object.

Table 7.94 - AttributeSet class

Description: Represents a set of attributes or parameters				
Derived From: IO_IdentifiedObject [ISO19111]				
Attributes:				
attrs	Attribute	O	N	Set of attributes that is contained in this attribute set

Table 7.95 - Ability class

Description: Describes module ability
Derived From: AttributeSet
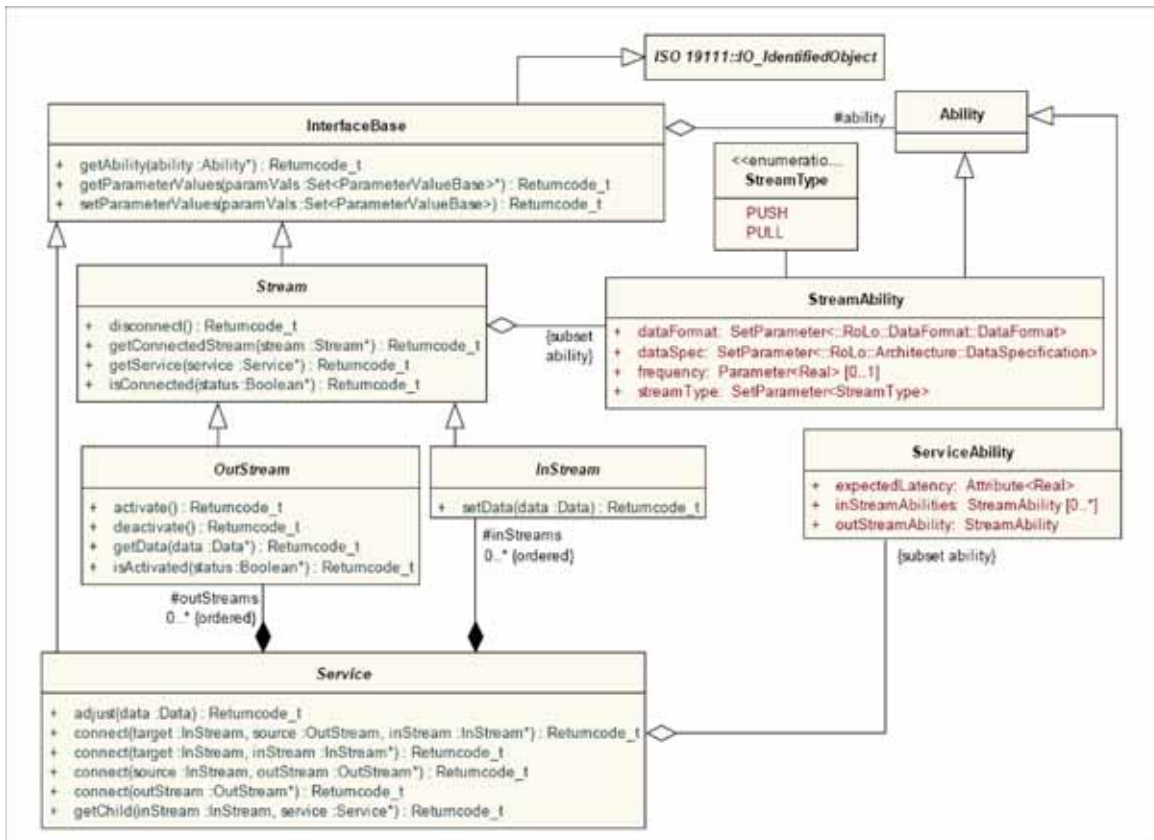
Figure 7.21 - RoLo Service

Table 7.96 - InterfaceBase class

Description:	Abstract class for interfacing objects			
Derived From:	IO_IdentifiedObject [ISO19111]			
Attributes:				
ability (protected)	Ability	M	1	Reference to an ability description for this object. The referred RoLo ability's attribute 'target' shall refer to this object.
Operations:				
getAbility	Operation for obtaining the ability description for this stream			
out / ability	Ability		Ability description of this stream	
setParameterValues	Operation for setting values to the configurable parameters			
in / paramVals	Set<ParameterValueBase>		Set of parameter values to be set. If some nonexistent or inconfigurable parameters were specified, UNSUPPORTED_PARAMETER or BAD_PARAMETER will be returned respectively.	
getParameterValues	Operation for obtaining status of configurable parameters			
out / paramVals	Set<ParameterValueBase>		Current status of parameter values	

Table 7.97 - StreamType enumeration

PUSH: Indicates that data passing is performed in PUSH mode (i.e., SENDER side triggers data passing).
PULL: Indicates that data passing is performed in PULL mode (i.e., RECEIVER side triggers data passing).

Table 7.98 - StreamAbility class

Description: Ability description for RoLo streams. If each RoLo stream has special functionalities, this class may be extended to be added to necessary descriptions.				
Derived From: Ability				
Attributes:				
dataSpec	SetParameter<RoLo::Architecture:: DataSpecification>	M	1	Set of RoLo data specifications that are supported by this stream.
dataFormat	SetParameter<RoLo::DataFormat:: DataFormat>	M	1	Set of RoLo data formats supported by this stream. When an instance of DataFormat::SpecificDataFormat is set by the user, the data specification specified in the data format (through SpecificDataFormat.dataSpec attribute) shall be consistent with the dataSpec attribute in this class.
streamType	SetParameter<StreamType>	M	1	Parameter for supported stream types
frequency	SetParameter<Real>	O	1	Parameter for data passing frequency in PUSH mode. The unit for this attribute is Hz. If unnecessary (for example, a RoLo out stream that only supports PULL type data passing), this parameter may be omitted.

Table 7.99 - Stream class

Description: Abstract class for representing RoLo streams		
Derived From: InterfaceBase		
Operations:		
getService	Returns the service owning this stream.	
out / service	Service	Reference to the service owning this stream
setConnectedStream	Obtain currently connected stream, if any.	
out / streams	Stream	Reference to the stream that is currently connected to this stream. If no stream is connected, ERROR is returned; otherwise, OK is returned. When the connection is performed without 'source' argument, this may not work (see description on Service class for details).
isConnected	Check whether this stream is connected to other stream.	
out / status	Boolean	If connected true, otherwise false.
disconnect	Disconnects this stream from the currently connected stream.	
Note – Values for the attribute 'ability' that is derived from parent class shall be limited to instances of StreamAbility or its inherited classes.		

Table 7.100 - OutStream class

Description:	Represents output streams	
Derived From: Stream		
Operations:		
getData	Obtain localization result	
out / data	Data (RoLo::Architecture)	Resulting localization data
activate	Activate stream output. Only meaningful on PUSH mode.	
deactivate	Deactivate stream output. Only meaningful on PUSH mode.	
isActivated	Query whether this stream is activated or not.	
out / status	Boolean	If activated true, otherwise false.

Table 7.101 - InStream class

Description:	Represents input streams		
Derived From: Stream			
Operations			
setData		Set data to this stream	
in	data	Data (RoLo::Architecture)	Localization data to be set to this stream.

Table 7.102 - ServiceAbility class

Description: Ability description for RoLo Service. If each specific service implementation has special functionalities, this class may be extended to be added to the necessary descriptions.				
Derived From: Ability				
Attributes:				
expectedLatency	Attribute<Real>	M	1	Expected latency. This ability descriptor is especially useful for Robotic Localization Service users. The unit for this attribute is milliseconds.
inStreamAbilities	StreamAbility	O	N	Ability descriptions for the input streams in this service.
outStreamAbility	StreamAbility	M	1	Ability descriptions for the output stream in this service.

Table 7.103 - Service class

Description: Interface for the robotic localization service					
Derived From: InterfaceBase					
Attributes:					
inStreams (protected)	InStream	O	N ord	An ordered list of RoLo input streams owned by this service.	
outStreams (protected)	OutStream	O	N ord	An ordered list of RoLo output streams owned by this service.	
connect	Establish connection from output stream to input stream. (SENDER service initiates the connection)				
in	target	InStream	Reference to a RoLo input stream to be connected. This target reference shall be obtained through getAbility method.		
in	source [0..1]	OutStream	Reference to the RoLo output stream that is connecting. This argument is optional. When this argument is omitted, 'getChildren' method may not work.		
out	inStream	InStream	Reference to a RoLo input stream to be used for further manipulation of the established connection. Note that, this reference may be pointing to a different object as the one given as input argument. Users shall use the returned reference, not the one obtained through getAbility method.		
connect	Establish connection from input stream to output stream. (RECEIVER service initiates the connection.)				
in	source [0..1]	InStream	Reference to the RoLo input stream that is connecting. This argument is optional. However, when data passing is to be done in PUSH mode, this argument cannot be omitted. Also, when this argument is omitted, 'getChildren' method may not work.		
out	out Stream	OutStream	Reference to a RoLo output stream object to be used for further manipulation of the established connection.		
adjust		Method for adjusting localization results. For elements not required for adjustment, don't-care element should be specified.			
in	data	Data(RoLo:: Architecture)	Data to be used for initialization or adjustment. Adjusts every element at once.		
getChild		Obtain services connected to input streams of this service.			
in	inStream	InStream	Instream to retrieve the connected service.		
out	services	List<Service>	Ordered list of services connected to the input streams of this service.		
Note – When 'getAbility' method is called, RoLo stream shall return an ability description that contains ability descriptors for the service and the descriptors for the RoLo streams that this service holds. This shall include the descriptors for each of the input streams. For the out stream, only a single descriptor is sufficient. Values for the attribute 'ability' that is derived from parent class shall be limited to instances of ServiceAbility or its inherited classes.					

7.6.1   Using RoLo Service

Here we show several non-mandatory steps and sequence diagrams as examples. Typical steps of using RoLo Services can be listed as follows:

1. (optional) Obtain ability description by calling 'getAbility' method toward RoLo service. An ability description obtained from RoLo service also includes descriptions on its streams. This step can be omitted if users already have sufficient information such by reading reference manuals.

2. (optional) Set up service and/or stream parameters through calling 'setParameterValues' method. If the default settings are sufficient or if there exists no parameter to be configured, this step can be omitted. In complicated cases, users may need to repeatedly call 'setParameterValues' and 'getParameterValues' to set and to confirm parameter changes.

3. Establish connection.

4. (optional) Set up initial position data by calling 'adjust' method with necessary data.

5. Perform data passing.

6. (optional) Occasionally, perform adjustment if necessary. Adjustment is an act to provide auxiliary information to the target module for improving the localization process.

7. Disconnect the connection.

Figures 7.22 to 7.26 show sequences of typical steps on using RoLo service. Note that in step 3, connection establishment can be initiated from two sides; either from the service that outputs data (SENDER service) or from the service that accepts data inputs (RECEIVER service). Figure 7.23 and Figure 7.24 show typical connection sequences in both cases. Note that, disconnection of the established connection (step 7) can be performed from both sides regardless of which side initiated the connection (Figure 7.26).
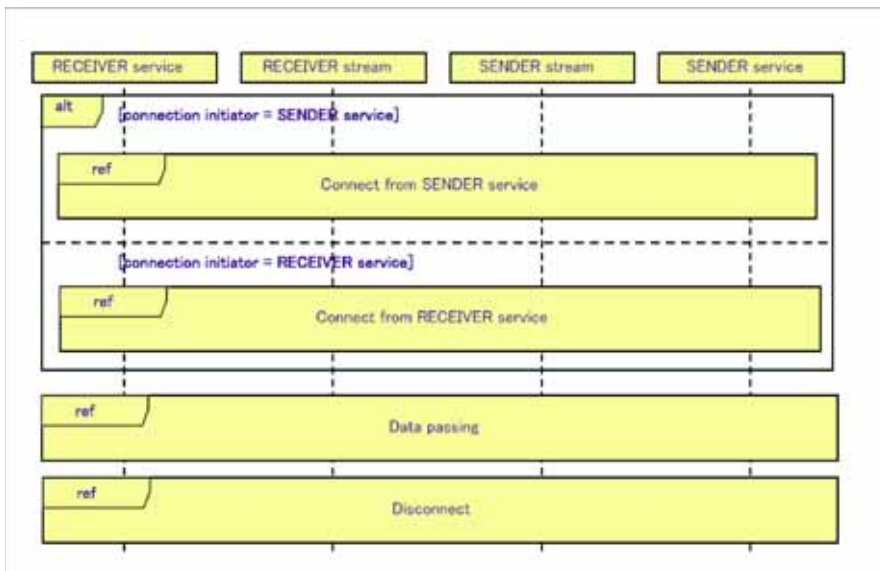
Figure 7.22 - Sequence diagram of typical RoLo service usage
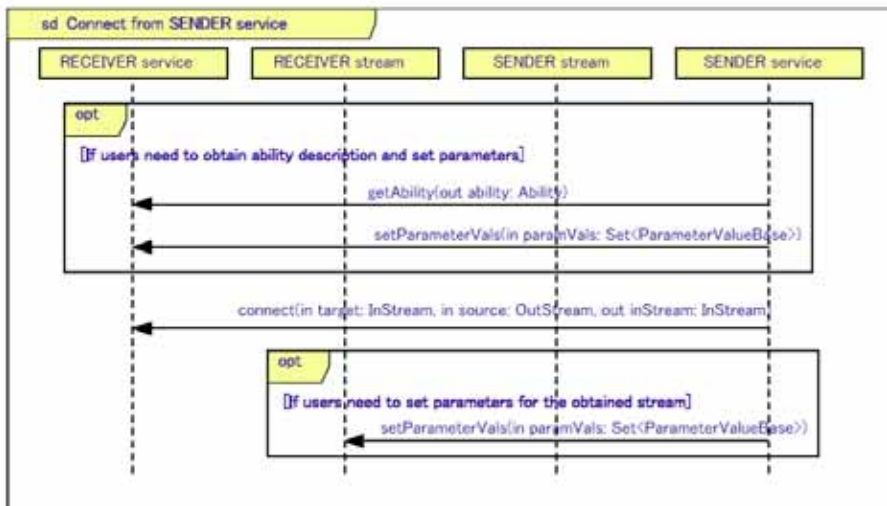
Figure 7.23 - Sequence Diagram of Connection Establishment from SENDER Service
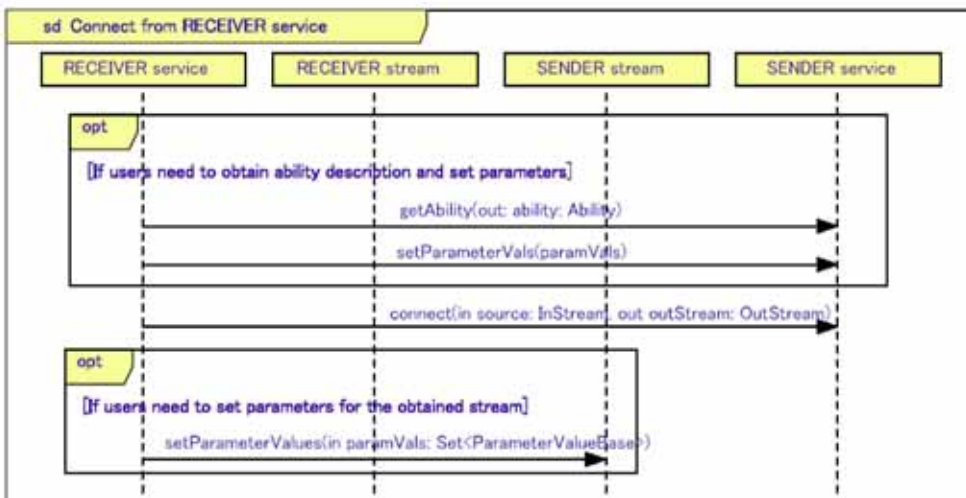
Figure 7.24 - Sequence Diagram of Connection Establishment from RECEIVER Service
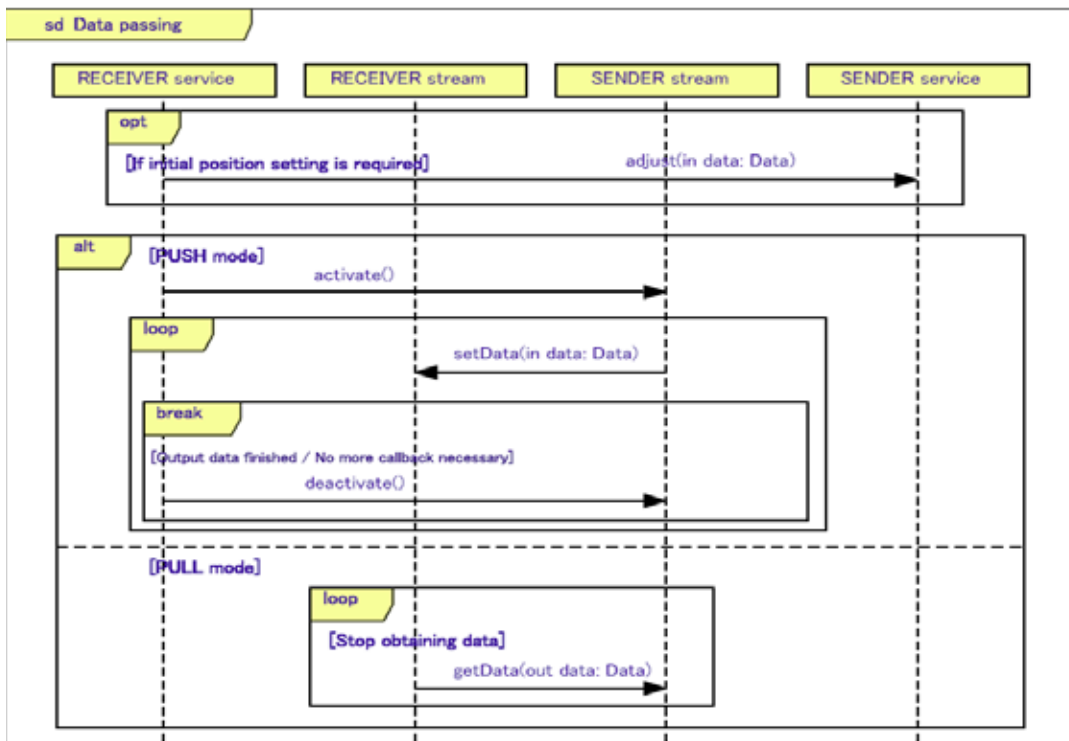
Figure 7.25 - Sequence Diagram of Data Passing
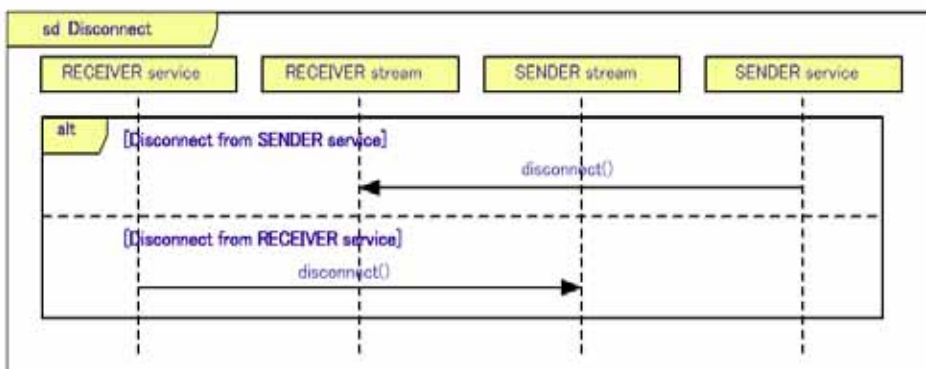
Figure 7.26 - Sequence Diagram of Disconnecting Connection

Another factor that needs consideration is the type of data passing. In this specification, two data passing types are provided as elements of StreamType enumeration: PUSH mode (SENDER side triggers data passing) and PULL mode (RECEIVER side triggers data passing). For example, most GPS receivers output data in PUSH mode, that is, measurement results are outputted continuously in some frequency. These two types of data passing can be performed regardless of which side initiates connection, as far as both modules have the ability to perform data passing in the

specified type. Figure 7.25 shows typical steps for performing data passing for the two directions. As can be seen from the sequence, in PULL mode, the RECEIVER service triggers data passing by calling 'getData' method. And in PUSH mode, the SENDER service triggers data passing by 'setData' method.

PUSH type data passing can also be understood as a callback from SENDER side to RECEIVER side. Thus, when using PUSH mode and when connection is established from IN side, the 'source' argument cannot be omitted. Without this, the RoLo output stream on SENDER side cannot know where to make callbacks for data passing. However, when connection is established from SENDER side, this 'source' argument is not required for the sake of making callbacks, as the RoLo input stream is given back as an 'inStream' argument.

8   Platform Specific Model

8.1   C++ PSM

This clause shows a PSM in C++ language based on the PIM described in Clause 7. This PIM-PSM mapping is based on the following rules:

- The return values of methods are assumed to be mapped as exceptions. Thus, in this PSM, no explicit description is given.

- When methods had only a single 'out' argument, it was mapped as return value of the corresponding function.

- The 'in' arguments to methods were mapped as method arguments with the 'const' modifier.

- Arguments that were based on non-primitive types are passed by reference.

- An attribute or an argument that is marked to occur more than once and is marked as unordered is mapped to '::std::set.' If marked as ordered, it is mapped to '::std::vector.'

- When an attribute is shown as an aggregation or as a derived attribute, or when an argument indicates a reference to other object, it is mapped as a pointer.

- CharacterString is mapped as '::std::string.'

The following shows the resulting C++ header files.

```cpp
// $Id: Returncode_t.hpp,v 1.3 2009/06/20 06:18:43 nishio Exp $

#pragma once

namespace RoLo
{
  enum Returncode_t {
    OK,
    ERROR,
    BAD_PARAMETER,
    UNSUPPORTED_PARAMETER,
    UNSUPPORTED_OPERATION,
    TIMEOUT
  };

}
// $Id: Architecture.hpp,v 1.3 2009/06/20 06:18:42 nishio Exp $
#pragma once

#include <RLS/RelativeCRS.hpp>
#include <RLS/MobileCRS.hpp>
#include <RLS/MobileOperation.hpp>
#include <RLS/Identity.hpp>
#include <RLS/ErrorType.hpp>
#include <RLS/Error.hpp>
#include <RLS/RoLoArchitecture.hpp>
```

```cpp
#include <RLS/RoLoDataOperation . hpp>
// $Id: Pose.hpp,v 1.0 2011/05/01 00:00:00 nishio Exp $

#pragma once
#include <ISO19111/CS_CoordinateSystem. hpp>
#include <ISO19111/SC_CoordinateReferenceSystem. hpp>

namespace RoLo
{
  namespace Architecture
  {
    class PoseType
    {
    public:
      PoseType* baseType;
    };

    class PositionPoseCRS
      : public ::ISO19111::SC_SingleCRS
    {
    public:
      PoseType* poseType;
      ::ISO19111: :SC_CRS positionBaseCRS;
    };

  }
}
// $Id: RelativeCRS.hpp,v 1.8 2009/06/20 17:51:30 nishio Exp $


#pragma once
#include <ISO19111/CS_CoordinateSystem.hpp>
#include <ISO19111/CD_Datum.hpp>
#include <RLS/RelativeCRS.hpp>
#include <RLS/Service.hpp>

namespace RoLo
{
  namespace Architecture
  {
    class MobileCRS
      : public DynamicRelativeCRS
    {
    };
```

```
        class MobileCartesianCRS
          : public MobileCRS
        {
        };

        class MobilePolarCRS
          : public MobileCRS
        {
        };

        class MobileDatum
          : public DynamicRelativeDatum
        {
        public:
          const ::RoLo::Interface::InStream& getInStream();
        protected:
          ::RoLo::Interface::InStream inStream;
        };

    }
}
// $Id: MobileOperation.hpp,v 1.5 2009/06/20 06:18:43 nishio Exp $

#pragma once
#include <ISO19111/CC_Operation.hpp>
#include <ISO19111/SC_CoordinateReferenceSystem.hpp>
#include <ISO19111/CD_Datum.hpp>

namespace RoLo
{
  namespace Architecture
  {
    class MobileOperation
      : public ::ISO19111::CC_Transformation
    {
    };

    class Mobile2StaticOperation
      : public MobileOperation
    {
    public:
      MobileCRS *source;
      ISO19111::SC_CRS *target;
    };

    class Staic2MobileOperation
      : public MobileOperation
    {
    public:
      ISO19111::SC_CRS *source;
      MobileCRS *target;
```

```cpp
    };

    class Mobile2MobileOperation
      : public MobileOperation
    {
    public:
      MobileCRS *source, *target;
    };

  }
}
// $Id: Identity.hpp,2011/05/01 00:00:00 nishio Exp $

#pragma once
#include <string>
#include <vector>
#include <set>
#include <ISO19111/IO_IdentifiedObject.hpp>
#include <ISO19111/CS_CoordinateSystem.hpp>
#include <ISO19111/SC_CoordinateReferenceSystem.hpp>
#include <ISO19111/CD_Datum.hpp>

name space RoLo
{
  namespace Architecture
  {
    class IdentityCS
      : public : :ISO19111: :CS_CoordinateSystem
    {
    };

    class NumericIdentityCS
    : public IdentityCS
    {
    };

    class SymbolicIdentityCS
    : public IdentityCS
    {
    };

    class IdentityDatum
      : public ::ISO19111::CD _Datum
    {
    };
```

```cpp
 class IdentityCRS
    : public ::ISO19111::SC_SingleCRS
 {
 };


 class NumericIdentityCRS
  : public IdentityCRS
};


 class SymbolicIdentityCRS
  : public IdentityCRS
{
};

class DirectSymbol
{
public:
   ::std: :vector<std: :string> coords; SymbolicIdentityCRS *crs;
};

class SymbolRef
{
public:
   DirectSymbol *point;
};

class SymbolicPosition {
public:
   DirectSymbol *direct; SymbolRef *indirect;
};

template <typename T>
class SetCoordinateSystemAxis
   : public : :ISO19111: :CS_CoordinateSystemAxis
{
public:
   ::std:set<T> elements;
};

class StringSetCoordinateSystemAxis
   : public SetCoordinateSystemAxis<std: :string>
{
```

```cpp
  };

  class StringSetCoordinateSystemAxis
      : public SetCoordinateSystemAxis<int>


// $Id: ErrorType.hpp,v 1.4 2009/06/20 06:18:43 nishio Exp $
#pragma once
#pragma once
#include <ISO19111/IO_IdentifiedObject . hpp>

name space RoLo
{
  namespace Architecture
  {
    class ErrorType
        : public : :ISO19111: :IO_IdentifiedObject {

    public:
      ErrorType *baseType;
    };

    class ErrorTypeOperation
        : public : :ISO19111: :IO_IdentifiedObject
    {
    public:
      ErrorType *source, *target;
    };

  }
}
// $Id: ErrorBase.hpp,v 1.1 2009/06/20 06:18:42 nishio Exp $

#pragma once
#include <RLS/ErrorType . hpp>

name space RoLo
{
  namespace Architecture {
    class ErrorType;
    class Error
        : public : :ISO19111: :IO_IdentifiedObjectBase
    {
    public:
```

```cpp
      ErrorType *errType;
    };

  }
}
// $Id: Error.hpp,v 1.8 2011/05/01 00:00:00 nishio Exp $

#pragma once
#include <vector>
#include <ISO19103/Primitive .hpp>
#include <ISO19111/IO_IdentifiedObject . hpp>
#include <RLS/ErrorType . hpp> #include <RLS/ErrorBase .
hpp> #include <RLS/RoLoArchitecture . hpp>

name space RoLo
{
  namespace Architecture
  {
    class Reliability
       : public Error, public ::ISO19103::Probability
    {
    };

    class ErrorDistribution : public Error
    {
    };

    class Matrix
    {
    public:
       int nRow, nCol;
       ::std::vector< ::ISO19103::Number > vals;
    };

    class CovarianceMatrix : public Matrix

     class Gaussian
     : public ErrorDistribution
    {
    public:
    CovarianceMatrix cov;
    };
```

```cpp
class UniformGaussian
: public Gaussian
{
};

class MixtureModel
  : public ErrorDistribution
{
};

class Model
  : public PositionElement
{
};

class WeightedModel
  : public Model
{
public:
  ::ISO19103: :Probability weight;
};

class LinearMixtureModel
  : public MixtureModel
{
public:
  ::std: :vector<Model> models;
};

class MixtureOfGaussian
  : public LinearMixtureModel

    class ParticleSet
      : public LinearMixtureModel
    {
    };

  }
}
// $Id: RoLoArchitecture.hpp,v 1.8 2009/06/20 06:18:43 nishio Exp $

#pragma once
```

```cpp
#include <vector>
#include <ISO19107/CoordinateGeometry . hpp> #include <ISO19111/
IO_IdentifiedObject . hpp> #include <ISO19111/
CS_CoordinateSystem. hpp>
#include <ISO19111/SC_CoordinateReferenceSystem. hpp> #include <RLS/
ErrorType . hpp>
#include <RLS/ErrorBase . hpp>
#include <RLS/Identity . hpp>

name space RoLo
{
  namespace Architecture
  {
    class Position
       : public ISO19111: :IO_IdentifiedObjectBase
    {
    public:
       SymbolicPosition* symbolic;
       ISO19107: :GM_Position* numeric;
    };

    class ElementSpecification
       : public : :ISO19111: :IO_IdentifiedObject
    {
    };

    class PositionElementSpecification
    : public ElementSpecification
    {
    public:
       ::ISO19111: :SC_CRS * crs; ErrorType
       *errType;
    };

    class ErrorElementSpecification : public
     ElementSpecification {
    public:
       ::std: :vector<PositionElementSpecification*> posSpecRefs; ErrorType *errType;
    };

    class Element
       : public : :ISO19111: :IO_IdentifiedObjectBase
```

```cpp
{
public:
   ElementSpecification *spec;
};


class PositionElement
: public Element
{
public:
   Position pos;
   Error err;
};

class ErrorElement : public Element
 {
public:
   Error err;
};

class DataSpecification
   : public : :ISO19111: :IO_IdentifiedObject
{
public:
   ::std: :vector<ElementSpecification> elemSpecs;
};

class Data
```

```cpp
        : public : :ISO19111: :IO_IdentifiedObjectBase
    {
    public:
        DataSpecification *spec;

        ::std: :vector<Element> elems;
    };

    class DontCare {
    };

    class NULLCS
        : public DontCare, public : : ISO19111: :CS_CoordinateSystem
    {
    };

    class NULLCRS
        : public DontCare, public ::ISO19111::SC _CRS
    {
    };

    class NULLDatum
        : public DontCare, public ::ISO19111::CD _Datum
    {
    };

    class NULLErrorType
        : public DontCare, public ErrorType
    {
    };

    class NULLElementSpecification
        : public DontCare, ElementSpecification
    {
    };
  }
}
// $Id: RoLoDataOperation.hpp,v 1.8 2009/06/20 17:46:15 nishio Exp $ #pragma once

#include <vector>
```

```cpp
#include <ISO19111/IO_IdentifiedObject . hpp> #include <ISO19111/
CC_Operation. hpp>
#include <RLS/RoLoArchitecture . hpp> name space RoLo

{
  namespace Architecture
  {
    class PositionElementOperation
      : public : :ISO19111: :IO_IdentifiedObject
    { public:
        PositionElementSpecification *source, *target;

    };
    class PositionElementConcatenatedOperation
      : public PositionElementOperation
    {
    public:
                            ::std: :vector<PositionElementOperation*> childOperations;
    };
    class PositionElementSingleOperation
      : public PositionElementOperation
    {
    public:
        ::ISO19111: :CC_CoordinateOperation *usesOperation; ErrorTypeOperation
        *usesErrTypeOperation;
    };
    class DataOperation
      : public : :ISO19111: :IO_IdentifiedObject
    {
    public:
        DataSpecification *source, *target;
    };
    class DataConcatenatedOperation
      : public DataOperation
    {
    public:
        ::std: :vector<DataOperation*> childOperations;
    };
    class DataSingleOperation
```

```cpp
            : public DataOperation
      {
      };

      class DataTransformation
        : public DataSingleOperation
      {
      public:
         ::std: :vector<PositionElementOperation*> usesOperations;
      };

      class DataMappingOperation
        : public DataSingleOperation
      {
      public:
              ::std: :vector<ElementSpecification*> sourceElemSpecs, targetElemSpecs;
      };

   }
}
// $Id: DataFormat.hpp,v 1.5 2009/06/20 06:18:42 nishio Exp $

#pragma once
#include <ISO19111/IO_IdentifiedObject . hpp>
#include <RLS/RoLoArchitecture . hpp>

name space RoLo
{
   namespace DataFormat
   {
      class DataFormat
        : public ISO19111: :IO_IdentifiedObject
      {
      };

      class EncodingRule
        : public DataFormat
      {
      };

      class SpecificDataFormat
        : public DataFormat
      {
      public:
         ::RoLo: :Architecture: :DataSpecification *dataSpec;
      };
```

```cpp
    class UserDefinedDataFormat
       : public SpecificDataFormat
    };

    class CommonDataFormat
       : public SpecificDataFormat
    {
    };

  }
}
// $Id: Interface.hpp,v 1.2 2009/06/20 06:18:43 nishio Exp $ #pragma once
                          #include <RLS/Ability . hpp>
                          #include <RLS/Service . hpp>


// $Id: Ability.hpp,v 2.0 2011/05/01 00:00:00 nishio Exp $

#pragma once
#include <set>
#include <ISO19103/Primitive .hpp>
#include <ISO19111/IO_IdentifiedObject . hpp> #include <ISO19115 .hpp>
#include <RLS/RoLoArchitecture . hpp>
#include <RLS/Error . hpp>

name space RoLo
{
  namespace Interface
  {
    class AttributeDefinition
       : public : :ISO19111: :IO_IdentifiedObject
    {
    public:
      ::ISO19115: :RS_Identifier type; ::ISO19103:
      :UnitOfMeasure unit;
    };

    class AttributeBase
       : public : :ISO19111: :IO_IdentifiedObject
    {
    public:
      const AttributeDefinition def;
    };

    template <typename T> class Attribute
       : public AttributeBase
```

```cpp
public:
  T val;
};

template <typename T> class Parameter
    : public Attribute<T>
{
public:
  bool isConfigurable;
};

template <typename T, typename TD> class
ParameterOverDomain
    : public Attribute<T>
{
public:
  TD domain;
};

template <typename T> class Interval
{
public:
  T min, max;
  bool minInc, maxInc;
};

template <typename T>
class IntervalParameter
    : public ParameterOverDomain< T, Interval<T> >
{};

template <typename T>
class SetParameter
    : public ParameterOverDomain< T, ::std::set<T> >
{};

class AttributeSet
    : public : :ISO19111: :IO_IdentifiedObject
{
public:
  ::std: :set<AttributeBase> attrs;
};

class Ability
    : public AttributeSet
```

```cpp
      };

      class ParameterValueBase {
      };

      template <typename T>
      class ParameterValue
         : public ParameterValueBase
      {
      public:
         T val;
      };
   }
}
// $Id: Service.hpp,v 1.11 2011/05/01 00:00:00 nishio Exp $

#pragma once
#include <vector>
#include <set>
#include <ISO19111/IO_IdentifiedObject . hpp> #include
<ISO19115 .hpp>
#include <RLS/Ability . hpp>
#include <RLS/DataFormat . hpp>
#include <RLS/RoLoArchitecture . hpp>

name space RoLo
{
   namespace Interface {
      enum StreamType { PUSH,
         PULL
      };

      class StreamAbility
         : public Ability
      {
      public:
         SetParameter< : :RoLo: :DataFormat: :DataFormat> dataFormat; SetParameter< :
         :RoLo: :Architecture: :DataSpecification > dataSpec; SetParameter<StreamType>
         streamType;
         SetParameter<double> frequency;
      };

      class InterfaceBase
         : public : :ISO19111: :IO_IdentifiedObject
      {
      public:
```

```cpp
    const Ability& getAbility();
    void setParameterValues(const : :std: :set<ParameterValueBase>& paramVals); const :
    :std: :set<ParameterValueBase>& getParameterValues();
protected:
  Ability* ability;
};

class Stream
    : public InterfaceBase
{
public:
  void disconnect();
  bool isConnected();
  const Stream& getConnectedStream(); const
  class Service& getService();
};

class OutStream
    : public Stream
{
public:
  const : :RoLo: :Architecture: :Data& getData(); void
  activate();
  void deactivate();
  bool isActivated();
};

class InStream
    : public Stream
{
public:
  void setData(const : :RoLo: :Architecture: :Data& data);
};

class ServiceAbility
    : public Ability
{
public:
  Attribute<int> maxOutStreamNum;
  Attribute<double> expectedLatency;
  ::std: :set<StreamAbility> inStreamAbilities;
  StreamAbility outStreamAbility;
};

class Service
    : public InterfaseBase
```

```
{
public:
        InStream& connect(const InStream& target, const OutStream* source = NULL);
   OutStream& connect (const InStream* source = NULL); void adjust(const : :RoLo:
   :Architecture: :Data& data); const : :std: :vector<const Service*> getChildren();

protected:
::std: :vector<InStream> inStreams; ::std: :vector<OutStream> outStreams;

     };

   }

}
```

Annex A
PSM for XML

(informative)

A.1 Overview

This annex provides a platform specific model for XML. This PSM has two variations: *generic model* and *architecture-specific* model. The *generic model* is derived by mapping naively from UML model of RoLo data to XML, and is able to represent any RLS elements. But, it is impossible to restrict structures syntactically for a specification of certain architecture even if the architecture of the data is known.

On the other hand, the *architecture-specific* model is generated for each RoLo specification in a pragmatic way, and is able to restrict its syntax strictly according to the specification. But the XML schema for the representation should be given for each RoLo data specification.

Hereafter, the target namespace of the given XML schemas is assumed to be "http://www.omg.org/spec/RLS/1 .1." Also, the prefix "RLS" indicates the same namespace.

A.2 Generic Model

The generic model is described below by XML schema definition.

```
<?xml version="1 .0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:rls="http://www.omg.org/spec/RLS/1.1"
    xmlns :gml="http: //www.opengis .net/gml/3 .2"
    targetNamespace="http://www.omg.org/rls/1.1"
    elementFormDefault="qualified" attributeFormDefault="qualified">

<xsd:import namespace="http://www.opengis.net/gml/3.2" schemaLocation="ISO_19136/ GML . xsd" />

    <!-- Fig.7 RoLo Error Type -->
    <xsd:element name="ErrorType" type="rls :ErrorTypeType"/>
    <xsd: complexType name="ErrorTypeType">
       <xsd: complexContent>
          <xsd:extension base="gml: IdentifiedObjectType">
             <xsd: sequence>
                 <xsd:element name="baseType" type="xsd: ID" minOccurs="0" maxOccurs="1"/>
          </xsd: sequence>
       </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<xsd:element name="ErrorTypeOperation" type="rls :ErrorTypeOperationType"/>
<xsd: complexType name="ErrorTypeOperationType">
   <xsd: complexContent>
      <xsd:extension base="gml: IdentifiedObjectType">
```

```
            <xsd: sequence>
                <xsd:element name="source" type="xsd: ID" minOccurs="1" maxOccurs="1"/>
                <xsd:element name="target" type="xsd: ID" minOccurs="1" maxOccurs="1"/>
            </xsd: sequence>
        </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<!-- Fig.8 RoLo Error -->
<xsd: element name="AbstractError" type="rls :AbstractErrorType" abstract="true" />
<xsd: complexType name="AbstractErrorType">
    <xsd: complexContent>
        <xsd:extension base="gml: IdentifiedObjectType">
            <xsd: sequence>
                <xsd:element name="errorType" type="xsd:ID" minOccurs="0" maxOccurs="1"/>
            </xsd: sequence>
        </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<!-- Reliability -->
<xsd:element name="Reliability" type="rls :ReliabilityType"
substitutionGroup="rls :AbstractError" />
    <xsd: complexType name="ReliabilityType">
        <xsd: complexContent>
            <xsd: extension base="rls :AbstractErrorType">
                <xsd: annotation>
                    <xsd: documentation>
                        xsd:restriction base="xsd:float"
                    </xsd: documentation>
                </xsd: annotation>
            </xsd: extension>
        </xsd: complexContent>
</xsd: complexType>

<!-- ErrorDistribution -->
<xsd:element name="AbstractErrorDistribution" type="rls :AbstractErrorDistributionType"
substitutionGroup="rls :AbstractError" abstract="true" />
    <xsd: complexType name="AbstractErrorDistributionType">
        <xsd: complexContent>
            <xsd: extension base="rls :AbstractErrorType" />
    </xsd: complexContent>
</xsd: complexType>

<!-- Gaussian -->
<xsd:element name="Gaussian" type="rls:GaussianType"
substitutionGroup="rls :AbstractErrorDistribution" />
    <xsd: complexType name="GaussianType">
        <xsd: complexContent>
            <xsd:extension base="rls :AbstractErrorDistributionType">
                <xsd: sequence>
                    <xsd:element name="cov" type="rls:CovarianceMatrixType" minOccurs="1"
maxOccurs=" 1" />
                </xsd: sequence>
            </xsd: extension>
        </xsd: complexContent>
```

```
    </xsd: complexType>

<xsd: complexType name="MatrixType">
    <xsd: simpleContent>
        <xsd : extension base="gml : doubleList">
            <xsd:attribute name="nRow" type="xsd:positiveInteger" use="required"/>
                <xsd:attribute name="nCol" type="xsd:positiveInteger" use="required"/>
        </xsd: extension>
    </xsd: simpleContent>
</xsd: complexType>

<xsd: complexType name="CovarianceMatrixType">
    <xsd: complexContent>
        <xsd:restriction base="rls:MatrixType">
            <xsd: annotation>
                <xsd: documentation>
                    Attributes "nRow" should be equal to "nCol"
            </xsd: documentation>
        </xsd: annotation>
    </xsd: restriction>
</xsd: complexContent> </xsd: complexType>

<!-- Uniform Gaussian -->
<xsd:element name="UniformGaussian" type="rls :UniformGaussianType"
substitutionGroup="rls :Gaussian" />
<xsd: complexType name="UniformGaussianType">
    <xsd: complexContent>
        <xsd:extension base="rls :GaussianType">
            <xsd: annotation>
                <xsd: documentation>
                    Attributes "nRow" and "nCol" should be "1".
                </xsd: documentation>
            </xsd: annotation>
        </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<!-- Mixture Model -->
<xsd:element name="AbstractMixtureModel" type="rls :AbstractMixtureModelType"
substitutionGroup="rls :AbstractErrorDistribution" abstract="true" />
    <xsd: complexType name="AbstractMixtureModelType" abstract="true">
        <xsd: complexContent>
            <xsd:extension base="rls :AbstractErrorDistributionType"/>
        </xsd: complexContent>
</xsd: complexType>

<!-- Linear Mixture Model -->
<xsd:element name="AbstractLinearMixtureModel" type="rls :AbstractLinearMixtureModelType"
substitutionGroup="rls :AbstractMixtureModel" abstract="true" />
    <xsd: complexType name="AbstractLinearMixtureModelType">
        <xsd: complexContent>
            <xsd:extension base="rls :AbstractMixtureModelType">
                <xsd: sequence>
                    <xsd:element name="models" type="rls:ModelType" minOccurs="0"
maxOccurs="unbounded" /> </xsd: sequence>
                </xsd: extension>
        </xsd: complexContent>
```

```
    </xsd: complexType>

<xsd: complexType name="ModelType">
<xsd:extension base="rls : PositionElementType"/>
</xsd: complexType>

<xsd: complexType name="WeightedModelType">
    <xsd:extension base="rls :ModelType">
        <xsd: sequence>
            <xsd:element name="weight" type="xsd:float" minOccurs="1" maxOccurs="1" />
        </xsd: sequence>
    </xsd: extension>
</xsd: complexType>

<!-- Particle Set -->
<xsd:element name="ParticleSet" type="rls :ParticleSetType"
substitutionGroup="rls :AbstractLinearMixtureModel" />
    <xsd: complexType name="ParticleSetType">
        <xsd: complexContent>
            <xsd:extension base="rls :AbstractLinearMixtureModelType">
                <xsd: annotation>
                    <xsd: documentation>
                        Each "model" element shall contain without "err" element.
                        This is interpreted that the error is a Gaussian distribution with
                        an all- zero covariance matrix.
                    </xsd: documentation>
                </xsd: annotation>
            </xsd: extension>
        </xsd: complexContent>
    </xsd: complexType>

<!-- MixtureOfGaussian -->
<xsd:element name="MixtureOfGaussian" type="rls:MixtureOfGaussianType"
substitutionGroup="rls :AbstractLinearMixtureModel" />
    <xsd: complexType name="MixtureOfGaussianType">
        <xsd: complexContent>
            <xsd:extension base="rls :AbstractLinearMixtureModelType">
                <xsd: annotation>
                    <xsd: documentation>
                        Each "model" element shall contain an error information of Gaussian
                        distribution.
                    </xsd: documentation>
                </xsd: annotation>
            </xsd: extension>
        </xsd: complexContent>
    </xsd: complexType>

<!-- Fig.10 RoLo Architecture -->
<!-- DataSpecification -->
<xsd:element name="DataSpecification" type="rls :DataSpecificationType"/>
<xsd: complexType name="DataSpecificationType">
    <xsd: complexContent>
        <xsd:extension base="gml: IdentifiedObjectType">
            <xsd: sequence>
                <xsd: choice minOccurs="1" maxOccurs="unbounded">
                    <xsd: element ref="rls : PositionElementSpecification" />
                    <xsd:element ref="rls :ErrorElementSpecification" />
```

```
            </xsd: choice>
        </xsd: sequence>
    </xsd: extension>
</xsd: complexContent> </xsd: complexType>

<!-- Data -->
<xsd:element name="Data" type="rls :DataType"/>
<xsd: complexType name="DataType">
    <xsd: complexContent>
        <xsd:extension base="gml: IdentifiedObjectType">
            <xsd: sequence>
                <xsd:element name="spec" type="xsd:ID" minOccurs="0" maxOccurs="1"/>
                <xsd: choice minOccurs="1" maxOccurs="unbounded">
                    <xsd: element ref="rls : PositionElement" />
                    <xsd:element ref="rls:ErrorElement" />
                </xsd: choice>
            </xsd: sequence>
        </xsd: extension>
    </xsd: complexContent> </xsd: complexType>

<!-- PositionElementSpecification -->
<xsd:element name="PositionElementSpecification"
type="rls : PositionElementSpecificationType"/>
    <xsd: complexType name="PositionElementSpecificationType">
        <xsd: complexContent>
            <xsd:extension base="gml: IdentifiedObjectType">
                <xsd: sequence>
                    <xsd:element name="crs" type="xsd:ID" minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="errType" type="xsd: ID" minOccurs="0" maxOccurs="1"/>
                </xsd: sequence>
            </xsd: extension>
        </xsd: complexContent>
    </xsd: complexType>

<!-- ErrorElementSpecification -->
<xsd:element name="ErrorElementSpecification" type="rls :ErrorElementSpecificationType"/> <xsd:
complexType name="ErrorElementSpecificationType">
    <xsd: complexContent>
        <xsd:extension base="gml: IdentifiedObjectType">
            <xsd: sequence>
                <xsd:element name="posSpecRefs" type="rls :PositionElementSpecificationType"
                minOccurs="1" maxOccurs="unbounded"/>
                <xsd:element name="errType" type="xsd:ID" minOccurs="1" maxOccurs="1"/>
            </xsd: sequence>
        </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<!-- ErrorElement -->
<xsd:element name="ErrorElement" type="rls :ErrorElementType"/>
<xsd: complexType name="ErrorElementType">
    <xsd: complexContent>
        <xsd:extension base="gml: IdentifiedObjectType">
            <xsd: sequence>
                <xsd:element name="spec" type="xsd:ID" minOccurs="1" maxOccurs="1"/>
                <xsd:element name="err" type="rls:AbstractErrorType" minOccurs="1"
```

```xml
                 maxOccurs="1"/
           </xsd: sequence>
        </xsd: extension>
     </xsd: complexContent>
</xsd: complexType>

<!-- PositionElement -->
<xsd:element name="PositionElement" type="rls:PositionElementType"/>
<xsd: complexType name="PositionElementType">
   <xsd: complexContent>
      <xsd:extension base="gml: IdentifiedObjectType">
         <xsd: sequence>
            <xsd:element name="spec" type="xsd:ID" minOccurs="1" maxOccurs="1"/>
            <xsd:element name="err" type="rls:AbstractErrorType" minOccurs="0"
             maxOccurs="1"/
            <xsd:element name="pos" type="rls:PositionType" minOccurs="1" maxOccurs="1"/>
         </xsd: sequence>
      </xsd: extension>
   </xsd: complexContent>
</xsd: complexType>

<!-- Position -->
<xsd: complexType name="PositionType">
   <xsd: choice>
      <xsd:element ref="rls:SymbolicPosition"/>
      <xsd:element ref="rls:NumericPosition"/>
   </xsd: choice>
</xsd: complexType>

<!-- Numeric Position -->
<xsd:element name="NumericPosition" type="rls :NumericPositionType"/> <xsd: complexType
name="NumericPositionType">
   <xsd: complexContent>
      <xsd:extension base="gml: IdentifiedObjectType">
         <xsd:group ref="gml:geometricPositionGroup"/>
      </xsd: extension>
   </xsd: complexContent>
</xsd: complexType>

<!-- Figure 5: Identity Information -->
<!-- SetCoordinateSystemAxis -->
<xsd:element name="SetCoordinateSystemAxis" type="rls : SetCoordinateSystemAxisType"
substitutionGroup="gml :CoordinateSystemAxis"/>
<xsd: complexType name="SetCoordinateSystemAxisType">
   <xsd: complexContent>
      <xsd : extension base="gml : CoordinateSystemAxisType" />
   </xsd: complexContent>
</xsd: complexType>

<!-- StringSetCoordinateSystemAxis -->
<xsd:element name="StringSetCoordinateSystemAxis"
type="rls : StringSetCoordinateSystemAxisType"
substitutionGroup="rls : SetCoordinateSystemAxis"/>
<xsd: complexType name="StringSetCoordinateSystemAxisType">
   <xsd: complexContent>
      <xsd:extension base="rls : SetCoordinateSystemAxisType"/>
   </xsd: complexContent>
```

```
        </xsd: complexType>

<!-- IntegerSetCoordinateSystemAxis -->
<xsd:element name="IntegerSetCoordinateSystemAxis" type="rls:
IntegerSetCoordinateSystemAxisType"
substitutionGroup="rls : SetCoordinateSystemAxis"/>
<xsd: complexType name="IntegerSetCoordinateSystemAxisType">
    <xsd: complexContent>
       <xsd:extension base="rls : SetCoordinateSystemAxisType"/>
    </xsd: complexContent>
</xsd: complexType>

<!-- SymbolicPosition -->
<xsd:element name="SymbolicPosition" type="rls :SymbolicPositionType"/>
<xsd: complexType name="SymbolicPositionType">
    <xsd: complexContent>
       <xsd: sequence>
           <xsd:element name="coords" type="rls:SymbolicCoordinateType" minOccurs="1"
            maxOccurs="unbounded" />
           <xsd:element ref="rls:SymbolicIdentityCRS" minOccurs="0" maxOccurs="1"/>
       </xsd: sequence>
    </xsd: complexContent>
</xsd: complexType>

<xsd: complexType name="SymbolicCoordinateType">
    <xsd: simpleContent>
       <xsd:extension base="xsd:string">
           <xsd:attribute name="axisName" type="xsd: string" use="required" />
       </xsd: extension>
    </xsd: simpleContent>
</xsd: complexType>

<!-- IdentityDatum -->
<xsd:element name="IdentityDatum" type="rls : IdentityDatumType"
substitutionGroup="gml :AbstractDatum"/>
    <xsd: complexType name="IdentityDatumType">
       <xsd: complexContent>
           <xsd:extension base="gml :AbstractDatumType"/>
       </xsd: complexContent>
    </xsd: complexType>

<!-- IdentityCRS -->
<xsd:element name="AbstractIdentityCRS" type="rls :AbstractIdentityCRSType" abstract="true"
substitutionGroup="gml :AbstractSingleCRS"/>
<xsd: complexType name="AbstractIdentityCRSType">
    <xsd: complexContent>
       <xsd: extension base="gml :AbstractCRSType">
           <xsd: sequence>
               <xsd:element name="usesDatum" type="xsd: ID" minOccurs="1" maxOccurs="1"/>
           </xsd: sequence>
       </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<!-- IdentityCS -->
<xsd:element name="AbstractIdentityCS" type="rls:AbstractIdentityCSType" abstract="true"
substitutionGroup="gml :AbstractCoordinateSystem"/>
```

```xml
<xsd: complexType name="AbstractIdentityCSType">
   <xsd: complexContent>
      <xsd:extension base="gml :AbstractCoordinateSystemType"/>
   </xsd: complexContent>
</xsd: complexType>

<!-- NumericIdentityCRS -->
<xsd:element name="NumericIdentityCRS" type="rls :NumericIdentityCRSType"
substitutionGroup="rls :AbstractIdentityCRS"/>
<xsd: complexType name="NumericIdentityCRSType">
   <xsd: complexContent>
      <xsd:extension base="rls :AbstractIdentityCRSType">
         <xsd: sequence>
            <xsd:element name="usesCS" type="xsd: ID" minOccurs="1" maxOccurs="1"/>
         </xsd: sequence>
      </xsd: extension>
   </xsd: complexContent>
</xsd: complexType>

<!-- NumericIdentityCS -->
<xsd:element name="NumericIdentityCS" type="rls :NumericIdentityCSType" substitutionGroup="rls
:AbstractIdentityCS"/>
<xsd: complexType name="NumericIdentityCSType">
   <xsd: complexContent>
      <xsd:extension base="rls :AbstractIdentityCSType"/>
   </xsd: complexContent>
</xsd: complexType>

<!-- SymbolicIdentityCRS -->
<xsd:element name="SymbolicIdentityCRS" type="rls :SymbolicIdentityCRSType"
substitutionGroup="rls :AbstractIdentityCRS"/>
<xsd: complexType name="SymbolicIdentityCRSType">
   <xsd: complexContent>
      <xsd:extension base="rls :AbstractIdentityCRSType">
         <xsd: sequence>
            <xsd:element name="usesCS" type="xsd: ID" minOccurs="1" maxOccurs="1"/>
         </xsd: sequence>
      </xsd: extension>
   </xsd: complexContent>
</xsd: complexType>

<!-- SymbolicIdentityCS -->
<xsd: element name=" SymbolicIdentityCS" type=" rls : SymbolicIdentityCSType" substitutionGroup="rls
:AbstractIdentityCS"/>
<xsd: complexType name="SymbolicIdentityCSType">
   <xsd: complexContent>
      <xsd:extension base="rls :AbstractIdentityCSType"/>
   </xsd: complexContent>
</xsd: complexType>

<!-- Pose -->
<!-- PoseType -->
<xsd:element name="PoseType" type="rls :PoseTypeType"/>
<xsd: complexType name="PoseTypeType">
   <xsd: complexContent>
      <xsd:extension base="gml: IdentifiedObjectType">
         <xsd: sequence>
```

```xml
                <xsd:element name="baseType" type="xsd: ID" minOccurs="0" maxOccurs="1"/>
            </xsd: sequence>
        </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<!-- PositionPoseCRS -->
<xsd:element name="AbstractPositionPoseCRS" type="rls :AbstractPositionPoseCRSType"
substitutionGroup="gml :AbstractReferenceSystemType"/>
<xsd: complexType name="AbstractPositionPoseCRSType">
    <xsd: complexContent>
        <xsd:extension base="gml :AbstractReferenceSystemType">
            <xsd: sequence>
                <xsd:element name="poseType" type="xsd: ID" minOccurs="1" maxOccurs="1"/>
                <xsd:element name="positionCRS" type="xsd:ID" minOccurs="1" maxOccurs="1"/>
            </xsd: sequence>
        </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<!-- Figure 3: Relative and Mobile coordinate reference system -->
<!-- RelativeDatum -->
<xsd:element name="AbstractRelativeDatum" type="rls :AbstractRelativeDatumType"
substitutionGroup="gml :EngineeringDatum"/>
<xsd: complexType name="AbstractRelativeDatumType">
    <xsd: complexContent>
        <xsd:extension base="gml :EngineeringDatumType"/>
    </xsd: complexContent>
</xsd: complexType>

<!-- StaticRelativeDatum -->
<xsd:element name="StaticRelativeDatum" type="rls :StaticRelativeDatumType"
substitutionGroup="rls :AbstractRelativeDatum"/>
<xsd: complexType name="StaticRelativeDatumType">
    <xsd: complexContent>
        <xsd:extension base="rls :AbstractRelativeDatumType">
            <xsd: sequence>
                <xsd:element name="dataSpec" type="xsd:ID" minOccurs="0" maxOccurs="1"/>
                <xsd:element name="base" type="rls:DataType" minOccurs="0" maxOccurs="1"/>
            </xsd: sequence>
        </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<!-- DynamicRelativeDatum -->
<xsd: element name="DynamicRelativeDatum" type="rls : DynamicRelativeDatumType"
substitutionGroup="rls :AbstractRelativeDatum"/>
<xsd: complexType name="DynamicRelativeDatumType">
    <xsd: complexContent>
        <xsd:extension base="rls :AbstractRelativeDatumType"/>
    </xsd: complexContent>
</xsd: complexType>

<!-- MobileDatum -->
<xsd:element name="MobileDatum" type="rls :MobileDatumType"
substitutionGroup="rls : DynamicRelativeDatum"/>
<xsd: complexType name="MobileDatumType">
```

```xml
        <xsd: complexContent>
            <xsd:extension base="rls :DynamicRelativeDatumType">
                <xsd: sequence>
                    <xsd:element name="inStream" type="xsd:ID" minOccurs="1" maxOccurs="1"/>
                </xsd: sequence>
            </xsd: extension>
        </xsd: complexContent>
    </xsd: complexType>

    <!-- RelativeCRS -->
    <xsd:element name="AbstractRelativeCRS" type="rls :AbstractRelativeCRSType"
    substitutionGroup="gml :EngineeringCRS"/>
    <xsd: complexType name="AbstractRelativeCRSType">
        <xsd: complexContent>
            <xsd:extension base="gml :EngineeringCRSType">
                <xsd: sequence>
                    <xsd:element name="usesDatum" type="xsd: ID" minOccurs="1" maxOccurs="1"/>
                </xsd: sequence>
            </xsd: extension>
        </xsd: complexContent>
    </xsd: complexType>

    <!-- StaticRelativeCRS -->
    <xsd:element name="AbstractStaticRelativeCRS" type="rls :AbstractStaticRelativeCRSType"
    substitutionGroup="rls :AbstractRelativeCRS"/>
    <xsd: complexType name="AbstractStaticRelativeCRSType">
        <xsd: complexContent>
            <xsd:extension base="rls :AbstractRelativeCRSType">
                <xsd: sequence>
                    <xsd:element name="usesDatum" type="xsd: ID" minOccurs="1" maxOccurs="1"/>
                </xsd: sequence>
            </xsd: extension>
        </xsd: complexContent>
    </xsd: complexType>

    <!-- StaticRelativeCartesianCRS -->
    <xsd:element name="StaticRelativeCartesianCRS" type="rls:StaticRelativeCartesianCRSType"
    substitutionGroup="rls :AbstractStaticRelativeCRS"/>
    <xsd: complexType name="StaticRelativeCartesianCRSType">
        <xsd: complexContent>
            <xsd:extension base="rls :AbstractStaticRelativeCRSType">
                <xsd: sequence>
                    <xsd:element name="usesCS" type="xsd:ID" minOccurs="1" maxOccurs="1"/>
                </xsd: sequence>
            </xsd: extension>
        </xsd: complexContent>
    </xsd: complexType>

    <!-- StaticRelativePolarCRS -->
    <xsd:element name="StaticRelativePolarCRS" type="rls :StaticRelativePolarCRSType"
    substitutionGroup="rls :AbstractStaticRelativeCRS"/>
    <xsd: complexType name="StaticRelativePolarCRSType">
        <xsd: complexContent>
            <xsd:extension base="rls :AbstractStaticRelativeCRSType">
                <xsd: sequence>
                    <xsd:element name="usesCS" type="xsd:ID" minOccurs="1" maxOccurs="1"/>
```

```xml
        </xsd: sequence>
      </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<!-- DynamicRelativeCRS -->
<xsd:element name="AbstractDynamicRelativeCRS" type="rls :AbstractDynamicRelativeCRSType"
substitutionGroup="rls :AbstractRelativeCRS"/>
<xsd: complexType name="AbstractDynamicRelativeCRSType">
    <xsd: complexContent>
       <xsd:extension base="rls :AbstractRelativeCRSType"/>
    </xsd: complexContent>
</xsd: complexType>

<!-- MobileCRS -->
<xsd:element name="AbstractMobileCRS" type="rls :AbstractMobileCRSType" substitutionGroup="rls
:AbstractDynamicRelativeCRS"/>
<xsd: complexType name="AbstractMobileCRSType">
    <xsd: complexContent>
       <xsd:extension base="rls :AbstractDynamicRelativeCRSType">
          <xsd: sequence>
             <xsd:element name="usesDatum" type="xsd: ID" minOccurs="1" maxOccurs="1"/>
          </xsd: sequence>
       </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<!-- MobileCartesianCRS -->
<xsd:element name="MobileCartesianCRS" type="rls :MobileCartesianCRSType" substitutionGroup="rls
:AbstractMobileCRS"/>
<xsd: complexType name="MobileCartesianCRSType">
    <xsd: complexContent>
       <xsd:extension base="rls :AbstractMobileCRSType">
          <xsd: sequence>
             <xsd:element name="usesCS" type="xsd: ID" minOccurs="1" maxOccurs="1"/>
          </xsd: sequence>
       </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<!-- MobilePolarCRS -->
<xsd:element name="MobilePolarCRS" type="rls :MobilePolarCRSType"
substitutionGroup="rls :AbstractMobileCRS"/>
<xsd: complexType name="MobilePolarCRSType">
    <xsd: complexContent>
       <xsd:extension base="rls :AbstractMobileCRSType">
          <xsd: sequence>
             <xsd:element name="usesCS" type="xsd: ID" minOccurs="1" maxOccurs="1"/>
          </xsd: sequence>
       </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>

<!-- Figure 13: RoLo Data Format -->
<!-- DataFormat -->
<xsd:element name="AbstractDataFormat" type="rls :AbstractDataFormatType"/>
<xsd: complexType name="AbstractDataFormatType">
```

```
    <xsd: complexContent>
        <xsd:extension base="gml: IdentifiedObjectType"/>
    </xsd: complexContent>
</xsd: complexType>

<!-- Figure 20 - RoLo Ability -->
<!-- AttributeSet -->
<xsd:element name="AttributeSet" type="rls :AttributeSetType"/>
<xsd: complexType name="AttributeSetType">
    <xsd: complexContent>
        <xsd:extension base="gml: IdentifiedObjectType"/>
    </xsd: complexContent>
</xsd: complexType>

<!-- Ability -->
<xsd:element name="Ability" type="rls:AbilityType" substitutionGroup="rls:AttributeSet"/>
<xsd: complexType name="AbilityType">
    <xsd: complexContent>
        <xsd:extension base="rls :AttributeSetType"/>
    </xsd: complexContent>
</xsd: complexType>

<!-- Figure 21 - RoLo Service --> <!-- StreamType -->
<xsd: simpleType name="StreamTypeType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="PUSH"/>
        <xsd:enumeration value="PULL"/>
    </xsd: restriction>
</xsd: simpleType>

<!-- StreamAbility -->
<xsd:element name="StreamAbility" type="rls :StreamAbilityType"
substitutionGroup="rls :Ability"/>
<xsd: complexType name="StreamAbilityType">
    <xsd: complexContent>
        <xsd:extension base="rls :AbilityType">
            <xsd: sequence>
                <xsd:element name="dataFormat" type="rls:AbstractDataFormatType" minOccurs="1"
                maxOccurs=" 1" />
                <xsd:element name="dataSpec" type="rls:DataSpecificationType" minOccurs="1"
                maxOccurs=" 1" />
            </xsd: sequence>
                <xsd:attribute name="streamType" use="required">
                    <xsd: simpleType>
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="PUSH"/>
                            <xsd:enumeration value="PULL"/>
                        </xsd: restriction>
                    </xsd: simpleType>
                </xsd: attribute>
                <xsd:attribute name="frequency" type="xsd:double" use="optional"/>
        </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>


<!-- ServiceAbility -->
```

```
<xsd:element name="ServiceAbility" type="rls :ServiceAbilityType"
substitutionGroup="rls :Ability"/>
<xsd: complexType name="ServiceAbilityType">
    <xsd: complexContent>
        <xsd:extension base="rls :AbilityType">
            <xsd: sequence>
                <xsd:element name="inStreamAbilities" type="rls:StreamAbilityType" minOccurs="0"
                maxOccurs="unbounded" />
                <xsd:element name="outStreamAbility" type="rls:StreamAbilityType" minOccurs="1"
                maxOccurs=" 1" />
            </xsd: sequence>
                <xsd:attribute name="expectedLatency" type="xsd:double" use="optional"/>
        </xsd: extension>
    </xsd: complexContent>
</xsd: complexType>
</xsd: schema>
```

A.3 Architecture-specific Model

While the generic model shown above can represent any RoLo data, it is redundant and over generalized so that it is difficult to check validity of the data syntactically according to the corresponding specifications. The architecture-specific model will provides another mapping of a RoLo data to XML that is tightly restricted for the corresponding RoLo architecture specifications.

A.3.1   Identifier and Tag Naming

In order to provide unique name of each component of RoLo data in a systematic way, we suppose that the following restrictions are applied to each related instance of RoLo architectures:

Each instance of DataSpecification, ElementSpecification, ErrorType, SymbolicIdentityCS, and ::ISO19111 :CS_CoordinateSystemAxis shall have an identifier attribute that follow the following syntax:

(In the following BNF, we use "<<" and ">>" instead of "<" and ">" to avoid confusion of XML's tags and nonterminal symbols.)

<<identifier>> : := <<namespace>> <<separator>> <<localname>>
<<namespace>> ::= <<xsd:anyURI>> <<separator>> ::= "/" | ":" | "#" <<localname>> ::= <<xsd::NCName>>

Here, <<xsd: anyURI>> and <<xsd: NCName>> are the restricted character strings that are defined in "W3CXML Schema Definition Language." From a given identifier that follows above syntax, we extract a namespace and a localname for a corresponding instance of Data, Element, Error, and SymbolicPosition and its axis name of coordinates using <<namespace>> and <<localname>>, respectively, part in <<identifier>>.

::ISO19111 :CS_CoordinateSystemAxis's axisAbbrev attribute shall identical to the <<localname>> part of its identifier attribute.

A.3.1.1 RoLo Data

Suppose that a DataSpecification has an identifier attribute, whose `<<name space>>` and `<<localname>>` part are "#myNamespace000" and "myRoLoData," respectively. We also suppose that the specification consists of a list of RoLo element specifications whose qualified names are "myElement0," "myElement1," "myElement2," and so on. Then, the XML schema of corresponding Data instance shall be as follows. Here we assume that the target namespace of the following schema is "#myNamespace000" that corresponds to "app" prefix.

```
<xsd: element name="myRoLoData" type="myRoLoDataType"/>
<xsd: complexType name="myRoLoDataType>
  <xsd: sequence>
    <xsd:element   ref="app:myElement0"/>
    <xsd:element   ref="app:myElement1"/>
    <xsd:element   ref="app:myElement2"/>
    ...
  </xsd: sequence>
</xsd: complexType>
```

Syntax of the contents of each Element is declared according to specifications of each ElementSpecification as described below.

Example

```
<app:SensedObjectInfo xmlns:app="#myApplication000">
   <app:id>
      <app:pos>
         <app:SensedObjectId srsName="#myCRS0003">
            <app:type>human</app:type>
            <app:color>red</app:color>
            <app:seqNum>0253</app:seqNum>
         </app:SensedObjectId>
      </app:pos>
      <app:err>
         <rls:Reliability>0.6</rls:Reliability>
      </app:err>
   </app:id>
   <app:location>
      <app:pos>
         <gml:Point srsName="#myCRS0005">
            <gml:pos>3.25 2.21</gml:pos>
         </gml:Point>
      </app:pos>
      <app:err>
         <rls:UniformGaussian>
         <rls:cov nRow="1" nCol="1">
            2.13
         </rls:cov>
         </rls:UniformGaussian>
      </app:err>
   </app:location>
      <app:time>
      <app:pos>
         <gml:TimeInstant frame="#myCRS0007">
            <gml:TimePosition>2009-01-01T00:40:00+09:00</gml:TimePosition>
```

```
            </gml:TimeInstant>
        </app:pos>
    </app:time>
</app:SensedObjectInfo>
```

A.3.1.2 PositionElement

Suppose that a PositionElementSpecification has an identifier attribute, whose namespace and localname part are "#myNamespace000" and "myPosElement," respectively. Then, the XML schema of a corresponding PositionElement shall be:

```
<xsd:element name="myPosElement" type="myPosElementType"/>
<xsd:complexType name="myPosElementType">
    <xsd:sequence>
        <xsd:element name="pos" type="<<myPosType>>"/>
        {<xsd:element name="err" type="rls:ErrorComponentType"/>}?
    </xsd:sequence>
</xsd:complexType>
, where <<myPosType>> is:
l an application specific SymbolicPosition type describe below if the
CS_CoordinateSystem of PositionElementSpecification refers an identityCS,
l gml:TimeInstantPropertyType, if the cs is a temporal coordiante system,
l or, gml:PointPropertyType otherwise.
```

The "err" element part can be omitted according to the specification.

Example 1

```
<app: id xmlns :app="#myApplication000">
    <app : pos>
        <app: SensedObjectID srsName="#myCRS0003">
            <app: type>human< /app : type>
            <app: color> red</app: color>
            <app: seqNum>0253</app: seqNum>
        </app : SensedObjectID>
    </app :pos>
    <app: err>
        <rls :Reliability>0 . 6</rls :Reliability>
    </app : err>
</ app : id>
```

Example 2

```
<app: location xmlns :app="#myApplication000"> <
        app : pos>
        <gml: Point srsName="#myCRS0005">
            <gml:pos>3.25 2.21</gml:pos>
        </gml : Point>
    </app :pos>
    <app: err>
        <rls :UniformGaussian>
            <rls:cov nRow="1" nCol="1">
                2.13
            </rls: cov>
        </rls :UniformGaussian>
    </app : err>
</app : location>
```

Example 3

```
<app:time xmlns :app="#myApplication000">
  <app:pos>
    <gml:TimeInstant frame="#myCRS0007"> <gml:TimePosition>2009-01-
      01T00:40:00+09:00</gml:TimePosition>
    </gml : TimeInstant>
  </app :pos>
</app : time>
```

A.3.1.3 ErrorElement

Suppose that a RoLo error element specification has an identifier attribute, whose namespace and localname parts are "#myNamespace000" and "myErrElement," respectively. Then, the XML expression of a corresponding *Error Element* shall be:

```
<xsd: element name="myErrElement" type="myErrElementType"/>
<xsd: complexType name="myErrElementType">
  <xsd: sequence>
    <xsd:element name="err" type="rls :ErrorComponentType"/> </
  xsd: sequence>
</xsd: complexType>
```

Example

```
<app:myError xmlns :app="#myApplication000">
  <app: err>
    <rls : Gaussian>
      <rls:cov nRow="3" nCol="1">
         2.31 -0.32   1.23
        -0.32         1.50 0.01
         1.23         0.01 10.31
      </rls: cov>
    </rls :Gaussian>
  </app : err>
</app :myError>
```

A.3.1.4 Symbolic Position

Suppose that an IdentityCS has an identifier attribute, whose namespace and localname parts are "#myNamespace000" and "myIdCS," respectively. We also suppose that the usesAxis attribute of the IdentityCS consists of a list of CoordinateSystemAxis [ISO19111] whose axisAbbrev (that is identical to the localname part in the identifier attribute of the axis) are "myAxis0," "myAxis1," "myAxis2," and so on. Then, the XML schema of a corresponding SymbolicPosition shall be as follows:

```
<xsd: element name="myIdCS" type="myIdCSType"/>
<xsd: complexType name="myIdCSType">
  <xsd: sequence>
                <xsd:element name="myAxis0" type="xsd:string"/>
                <xsd:element name="myAxis1" type="xsd:string"/>
                <xsd:element name="myAxis2" type="xsd:string"/>
      ...
  </xsd: sequence> </xsd: complexType>
```

Example

```
<app:id xmlns:app="#myApplication000" srsName="#myCRS0003">
    <app: type>human< /app : type>
    <app: color>red</app: color>
    <app: seqNum>0253</app: seqNum>
</app: id>
```

Annex B
Naming of RoLo Architecture Components for Filter Condition

(informative)

This annex provides a naming rule of RoLo architecture components for use with filter condition. In order to utilize the filter condition, we need a way to specify components in a RoLo data to test each condition. For this purpose, we suppose that each RoLo data can be expressed by XML-PSM (see Annex A) and use XPath to indicate each part of RoLo data as same as the original filter encoding in ISO 19143.

Example 1

This example is an XML encoding of a filter condition that requires only localization data in a certain area.

```
<fes:Filter
    xmlns:fes="http://www.opengis.net/fes/2.0"
        xmlns:gml="http://www.opengis.net/gml/3.1"
    xmlns:myapp="http://my.localhost.localnet/myapp"
    xmlns:rls="http://www.omg.org/rls/1.0">
    <fes:Intersects>
        <fes:PropertyName>myapp:location/rls:pos</fes:PropertyName>
        <gml:Envelope
            srsName="http://my.localhost.localnet/myapp/crs000">
                <gml:lowerCorner>10.25 15.33</gml:lowerCorner>
                <gml:upperCorner>17.73 25.03</gml:upperCorner>
        </gml:Envelope>
    </fes:Intersects>
</fes:Filter>
```

Example 2

This example is an XML encoding of a filter condition that requires only localization data of a certain ID.

```
<fes:Filter
    xmlns:fes="http://www.opengis.net/fes/2.0"
    xmlns:gml="http://www.opengis.net/gml/3.1"
    xmlns:myapp="http://my.localhost.localnet/myapp"
    xmlns:rls="http://www.omg.org/rls/1.0">
    <fes:PropertyIsEqualTo>
        <fes:PropertyName>myapp:id/rls:pos</fes:PropertyName>
        <fes:Literal>myID:3429:abcd</fes:Literal>
    </fes:PropertyIsEqualTo>
</fes:Filter>
```

Example 3

This example is an XML encoding of a filter condition that requires only localization data in a certain area and a certain time period.

```
<fes:Filter
    xmlns:fes="http://www.opengis.net/fes/2.0"
             xmlns:gml="http://www.opengis.net/gml/3.1"
    xmlns:myapp="http://my.localhost.localnet/myapp"
    xmlns:rls="http://www.omg.org/rls/1.0">
<fes:And>
    <fes:Intersects>
        <fes:PropertyName>myapp:location/rls:pos</fes:PropertyName>
        <gml:Polygon
                srsName="http://my.localhost.localnet/myapp/crs000">
            <gml:exterior>
                <gml:LinearRing>
                    <gml:posList dimension="2">
                        23.02 34.21
                        11.56 23.14
                        90.43 23.19
                        33.23 29.00
                        23.02 34.21
                    </gml:posList>
                </gml:LinearRing>
            </gml:exterior>
        </gml:Polygon>
    </fes:Intersects>
    <fes:PropertyIsBetween>
        <fes:PropertyName>myapp:time/rls:pos</fes:PropertyName>
            <fes:LowerBoundary>
                <fes:Literal>2008-12-08T09:00:00.000-08:00</fes:Literal>
                </fes:LowerBoundary>
            <fes:UpperBoundary>
                <fes:Literal>2008-12-10T17:30:00.000-08:00</fes:Literal>
            </fes:UpperBoundary>
        </fes:PropertyIsBetween>
    </fes:And>
</fes:Filter>
```