

# **SEDRIS: The Technology Components**

*<http://www.sedris.org>*

**SEDRIS™ Technology Conference**  
**Lake Buena Vista, FL**  
**6 January 2004**

**Jesse Campos**  
**[jesse.j.campos@saic.com](mailto:jesse.j.campos@saic.com)**



# About this Tutorial

- **DESCRIPTION**

- This tutorial introduces the five SEDRIS technology components (DRM, EDCS, SRM, API, STF) with an emphasis on the DRM. The role that each component plays within the environmental data domain is explored to provide a springboard for understanding the other tutorials and presentations in the conference. This tutorial highlights the most commonly used areas of the DRM, the key DRM classes, and how the EDCS and the SRM components are utilized within the DRM. An overview of the API's powerful capabilities, including its most commonly used functions, is also provided.

- **WHO SHOULD ATTEND**

- Software engineers, systems engineers, and environmental modelers new to SEDRIS technologies.

- **PREREQUISITE**

- A basic knowledge of environmental data modeling and use, an understanding of software development and APIs, and familiarity with UML notation is recommended.

- **WHAT TO EXPECT**

- At completion, the attendee will have an understanding of the five SEDRIS technology components, the basic elements provided within the DRM and API, and be better prepared to take advantage of the remaining conference tutorials and presentations.



## Prerequisites

---

- **To get the most from this tutorial, we assume you know the following information:**
  - **A basic knowledge environmental modeling techniques, common environmental data, and data representation approaches.**
  - **A basic understanding of object-oriented design techniques.**
  - **Understanding of software development and APIs.**
  - **Familiarity with UML notation.**



# Tutorial Outline

---

- **Environmental Data Challenge**
- **What is SEDRI?**
- **Using SEDRI Products**
- **Data Models and a Data Representation Model**
- **Environmental Data Coding Specification (EDCS)**
- **Spatial Reference Model (SRM)**
- **Data Representation Model (DRM)**
- **SEDRI Transmittal Format (STF)**
- **Application Program Interfaces (APIs)**
- **SEDRI Software Development Kits (SDKs)**
- **Conclusion**

# **Environmental Data Challenge**



- How much data is stored in one year?
  - \_\_\_\_\_ => \_\_\_\_\_ terabytes
  - \_\_\_\_\_ U.S. Libraries of Congress
- How much data was produced in one year?
  - 3.5X stored
- How much per person?
  - \_\_\_\_\_ MB
- Doubling every \_\_\_\_\_



- **Disparate sources of data**
  - Paper
  - Sensors
  - Models
  - Machines
  - Humans
- **Quality of Data is questionable**
- **Can the data be processed into information?**
- **Is the data**
  - Accessible
  - Usable
  - Trustworthy



# Environmental Data Challenges

---

- **Regardless of what format(s) or product(s) are utilized, users need to represent all of their environmental data in a unified manner such that changes, updates, modifications, or additions (to systems or sub-systems) can be handled in an efficient and methodical manner**
- **Some critical issues are:**
  - **How individual data items (objects) relate to each other**
  - **How new objects/capabilities/features can be added without fundamentally changing the underlying schema**
  - **How formats will be isolated from object-level representations or interfaces**





## **Environmental Data Challenges (cont'd)**

---

- **Representation of location for the various coordinate systems (spatial reference frames), local or global, that will be “natural” for individual systems or sub-systems**
- **Accurate, efficient, and fast conversion of location data between different spatial reference frames**
- **Comprehensive dictionary of terms that not only deals with terrain data, but also atmosphere, ocean, littoral, and space data. And is also extensible in a predictable and supported manner**
- **A representation schema that can handle any resolution, type, organization, and extent of environmental data through a uniform approach for all domains of the environment**



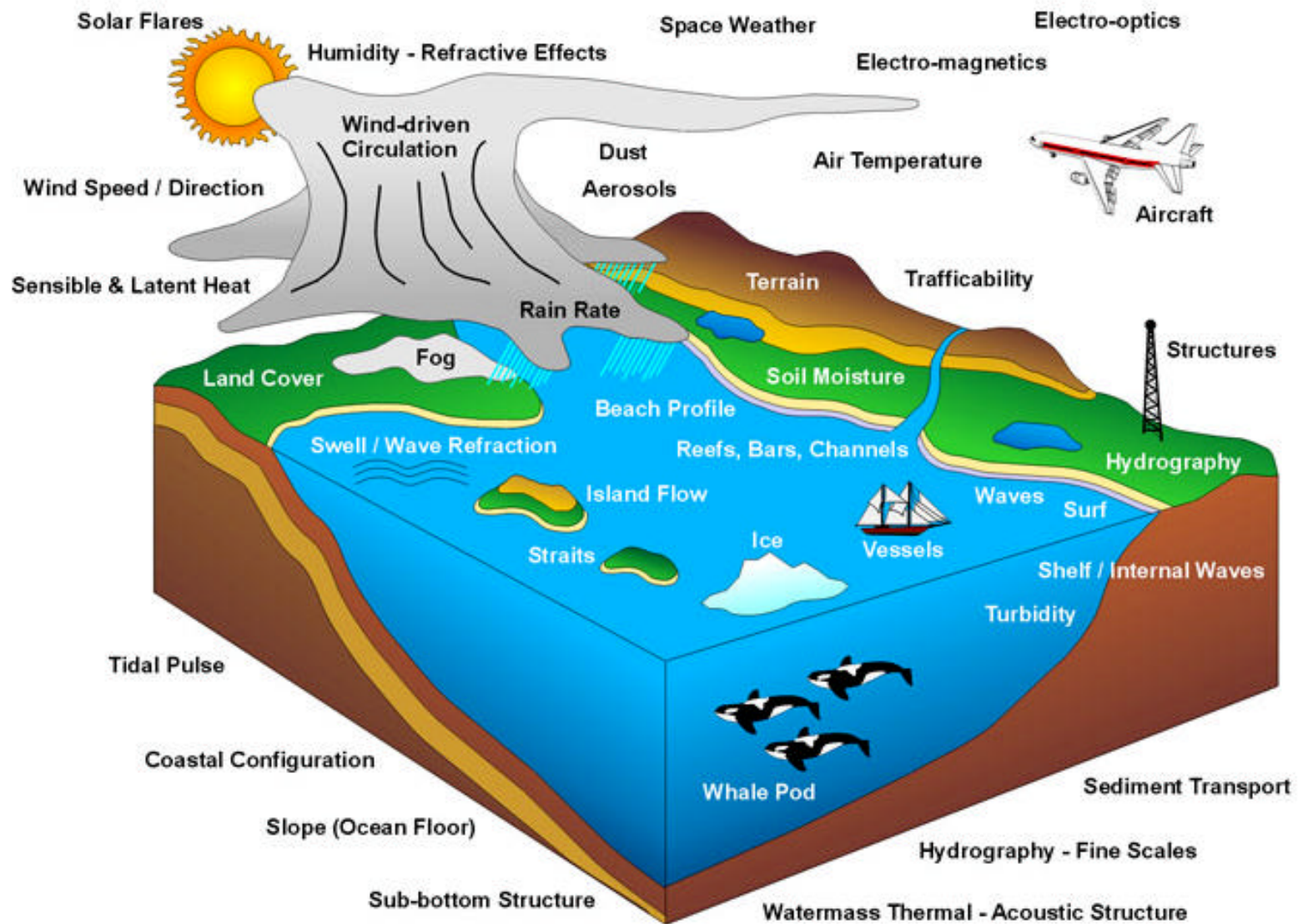
## **Environmental Data Challenges (cont'd)**

---

- **A mechanism to access and interact with any of the previous items through a robust software interface**
- **Capture and communicate the resulting data in a persistent, efficient, and platform independent format designed to handle large and distributed data sets**
- **Tools to manipulate, evaluate, visualize, or analyze the data**
- **Automatically evaluate and validate data sets against stated requirements**



# All Environmental Data



# **What is SEDRIS ?**



## **SEDRIIS Is ...**

---

- **A method for unambiguously describing the environment**
- **A mechanism to share and interchange the described environment**
- **Sharing Environmental Data Responsibly with an Interface Specification**
- **An infrastructure technology**
- **A way of thinking about environmental concepts and their representation**
- **Solution for Environmental Data Representation, Interchange, and Specification**



## Tangibly - SEDRIIS Is ...

---

- **A suite of software products**
  - Software Development Kits
  - Tools
  - Data sets
  - ISO Standards
- **An organization**
  - Managing the core software products
  - Providing education and support
- **Associates**
  - Users and developers of SEDRIIS products



# Primary Goals of SEDRI

---

- **Unambiguous representation of environmental data**
  - Semantics and relationships of data elements
  - All environmental domains
  - Expressed in a data representation model
- **Efficient interchange of environmental data**
  - Sharing and re-use
  - Ease of access and software development
  - Tools and applications



# SEDRIIS Technical Objectives

---

- **Articulate and capture the complete set of data elements and associated relationships needed to fully represent environmental data:**
  - **Data Representation Model (DRM)**
  - **Environmental Data Coding Specification (EDCS)**
  - **Spatial Reference Model (SRM)**
- **Support the full range of applications across all environmental domains (terrain, ocean, atmosphere, and space) and 3-D models of the physical environment.**
- **Support an integrated model of environmental representation**
- **Provide a standard interchange mechanism to distribute environmental data and promote database reuse among heterogeneous applications.**
  - **API**
  - **STF**





# Technology Components of SEDRIIS

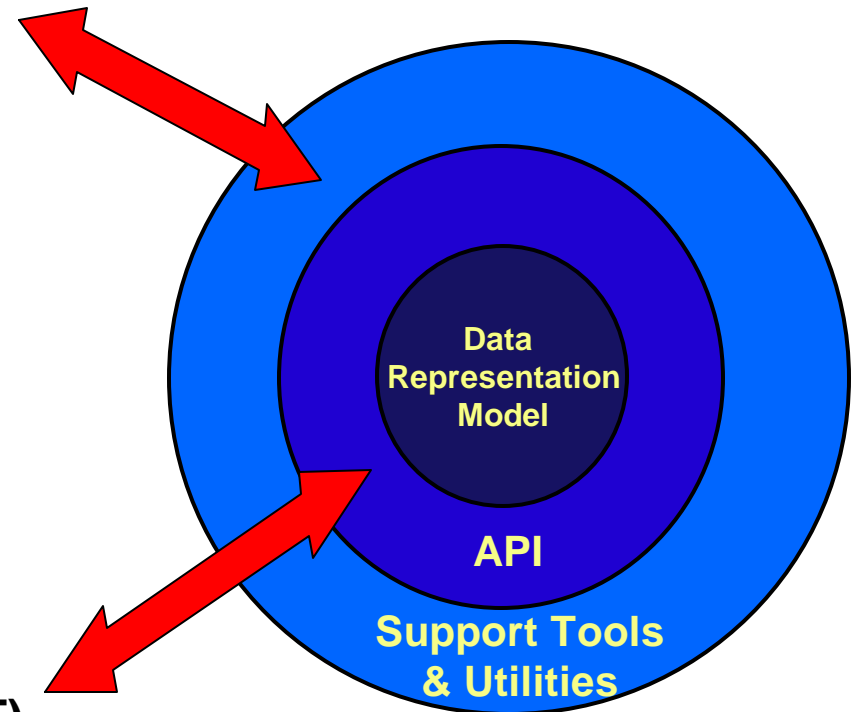
---

- **Spatial Reference Model (SRM):** Unified and robust description of the coordinate systems, along with an accurate, efficient, and fast software implementation
- **Environmental Data Coding Specification (EDCS):** Provides “thing” level semantics (the “dictionary” of the language) (classify/attribute scheme)
- **Data Representation Model (DRM):** Provides syntax and structural semantics for representing environmental data and databases (the “grammar” of the “language”)
- **SEDRIIS Application Program Interface(API):**
  - Allows ease of access
  - Lowers the barrier-to-entry in software development
  - Provides read, write, and modify capabilities
- **SEDRIIS Transmittal Format (STF):** Platform independent storage and transmission of data



# How SEDRIIS has been Developed

- **SEDRIIS Associates** (key environmental database developers/users)
  - Review and feedback
    - DRM
    - API
    - EDCS
    - SRM
  - Native-model mapping
  - Interchange experiments
  - Value-added tools/utilities
- **Core Team**
  - Manage evolution
    - DRM, API, EDCS, & SRM
  - Reference implementation(s)
  - SEDRIIS Transmittal Format (STF)
  - Common tools & applications





# SEDRIIS Associate Organizations

---

- *3D Pipeline*
- *Accent Geographic, Inc.*
- *AcuSoft, Inc.*
- *Advanced Interactive Systems (AIS), Inc.*
- *The Boeing Company*
- *CAE - Canada*
- *Charles River Analytics, Inc. (CRA)*
- *DataMat S.p.A. - Italy*
- *ERDAS, Inc.*
- *Evans and Sutherland (E&S)*
- *Indra - Spain*
- *JRM Technologies, Inc.*
- *L3 Communications - Link Simulation & Training*
- *Lockheed Martin Information Systems (LMIS)*
- *Lockheed Martin Tactical Defense Systems (LMTDS)*
- *MultiGen - Paradigm Inc. (MPI)*
- *Northrop Grumman Information Technology (NGIT)*
- *Object Raku Technology, Inc. - Canada*
- *OKTAL - France*
- *ProLogic, Inc.*
- *Raytheon Company Electronic Systems*
- *Rheinmetall Defence Electronics GmbH - Germany*
- *Science Applications International Corporation (SAIC)*
- *SGI*
- *Sogitec Industries S.A. - France*
- *Tenix Defence Pty. Ltd. - Australia*
- *Terrain Experts, Inc.*
- *TerraSim, Inc.*
- *Thales Training & Simulation (TT&S) – United Kingdom, France*
- *Vcom3D, Inc.*
- *Veridian*



## Where SEDRI is headed

---

- **SEDRI will remain a DMSO-sponsored project until all ISO/IEC standards are completed**
- **Then, the technologies will transition to another entity, based on the results of on-going evaluation of business approaches and impacts**
- **Entities being considered include (but not limited to): another DoD or government organization; existing consortia; existing commercial organizations; establishment of new organization; academia; ...)**
- **A fundamental criteria for transition is the guarantee to support existing and emerging customers and associates**
- **A number of business criteria and evaluation factors are being used**

# Using SEDRIS Products



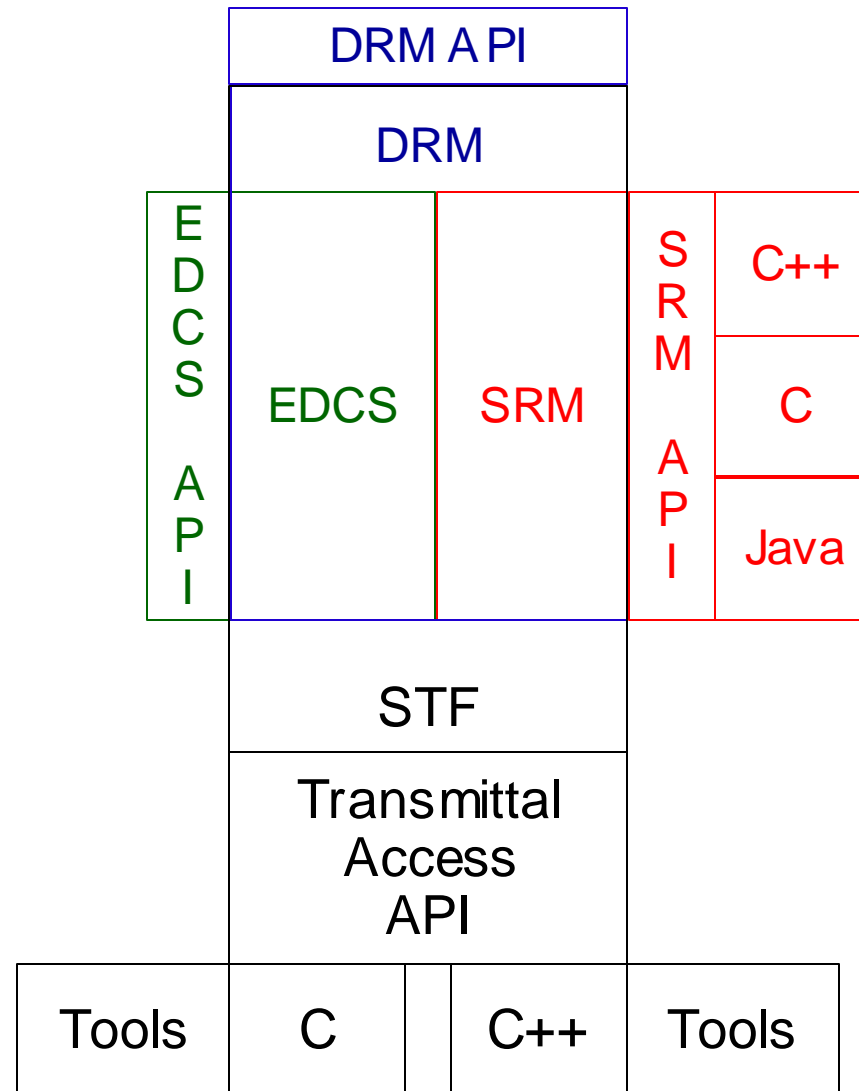
## **SEDRIIS can be used to build...**

---

- **A repository or a library system for environmental data**
- **An authoring tool or an environmental database generation system**
- **A specific environmental database**
- **An archiving or data discovery mechanism**
- **A scenario generation system**
- **An application that converts databases**

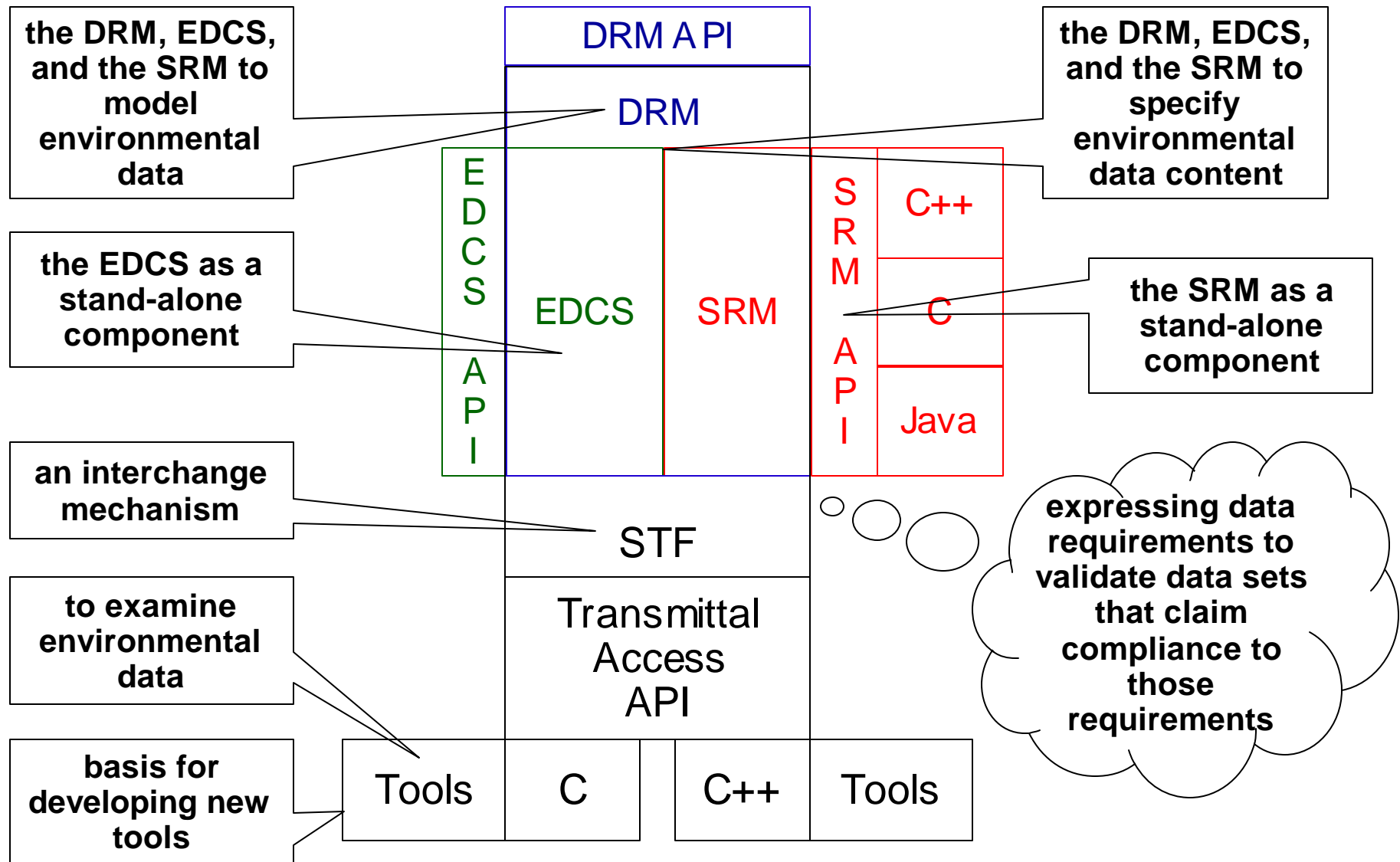


# SEDRIIS Components





# Using the SEDRIS Components







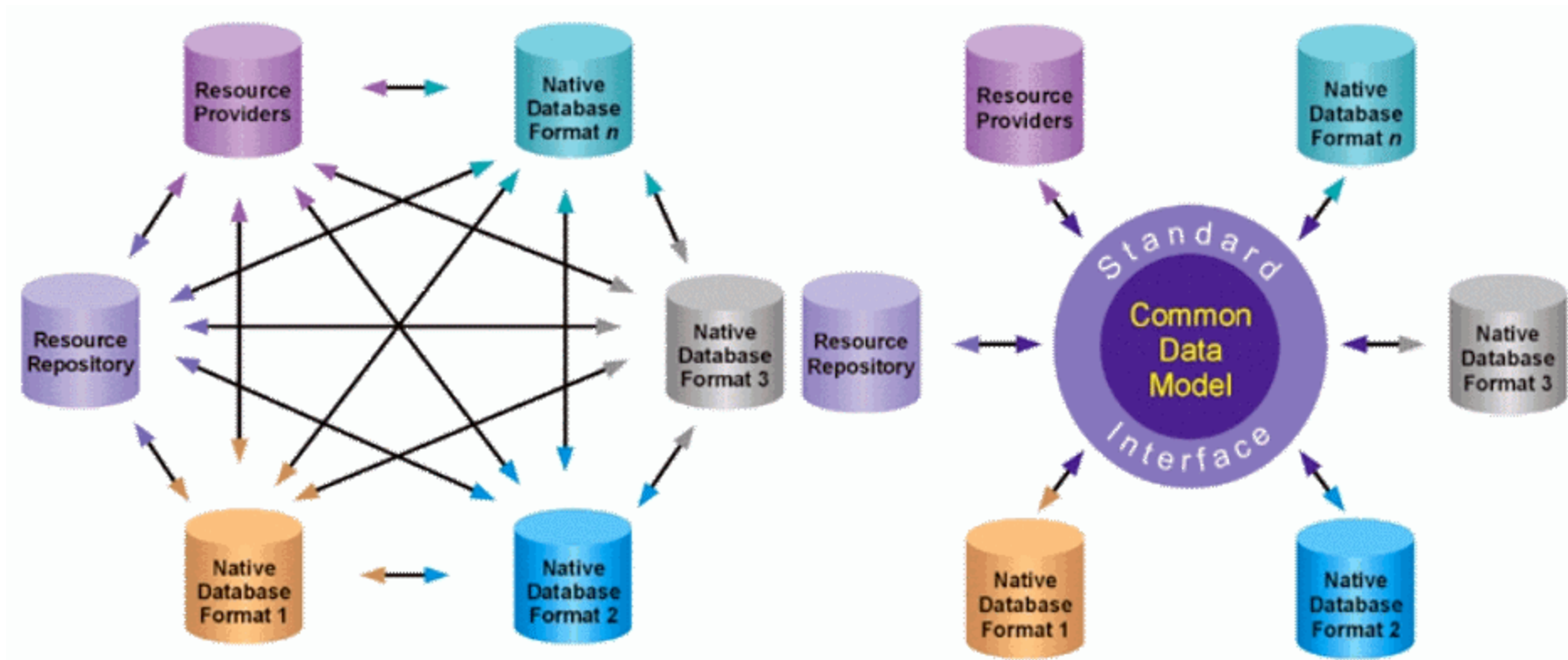
## How to Develop SEDRIIS Expertise

---

- **Learn to speak SEDRIIS: learn the DRM**
- **Generate mapping documents**
  - For native format(s) or assigned government format(s)
  - To ensure the DRM and EDCS can handle all data requirements
- **Develop software: to convert native data into SEDRIIS and back to check completeness of the interchange**
- **Become a SEDRIIS Associate and participate in SEDRIIS Associates Meetings (SAMs) and interchange experiments**
  - Exchange ideas
  - Cooperatively define and develop SEDRIIS technology
  - Share non-proprietary (native format) utilities and applications that support SEDRIIS interchange

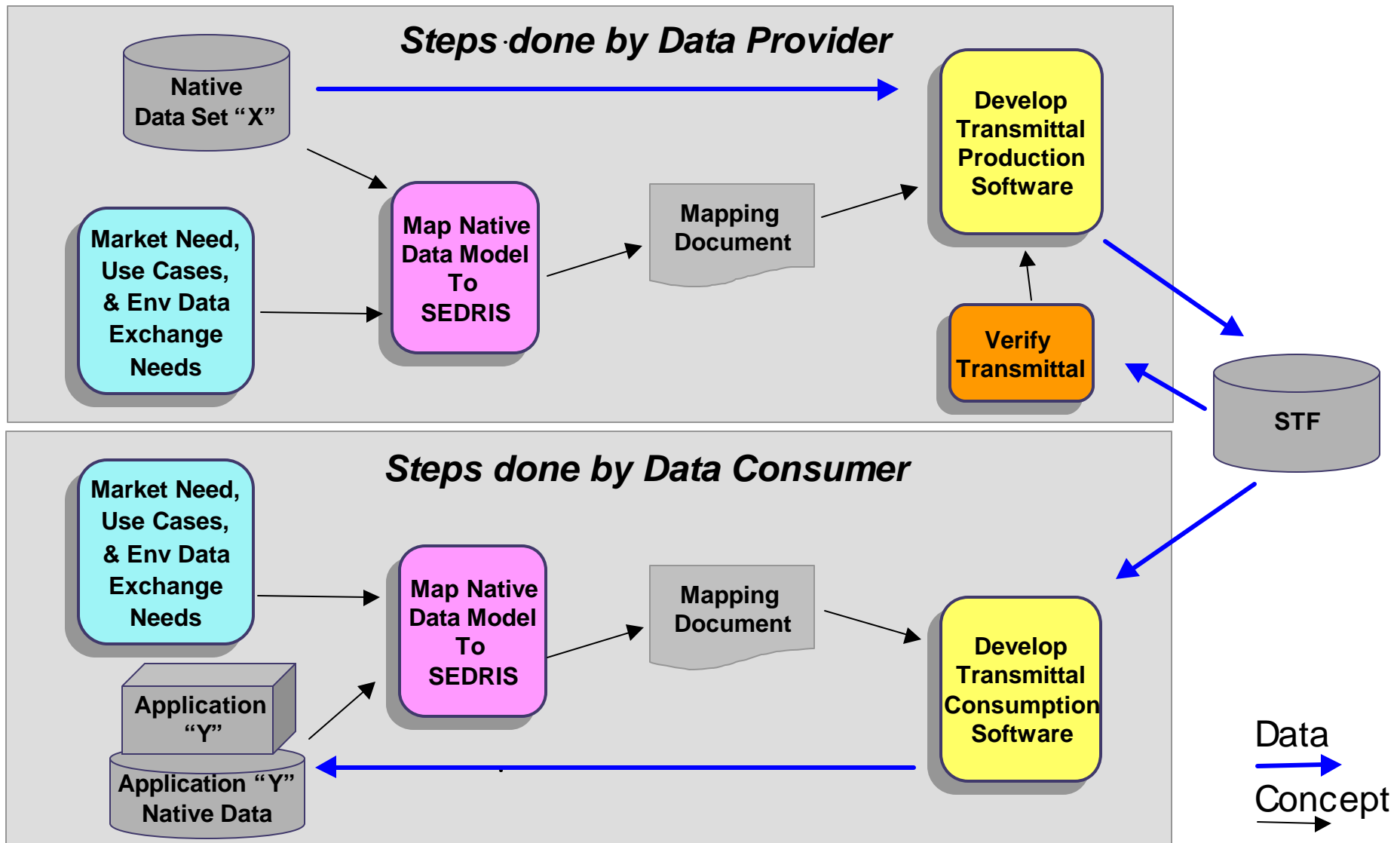


# The SEDRI Interchange Advantage





# Steps in SEDRIS Production and Consumption Process



# **Data Models and a Data Representation Model**



# Data Models in General

---

- **What is a data model?**
  - The specific set of data structures, and their relationships, that define a unique instance of data about a concept or an object.
  - **Example:**
    - A customer is represented by a customer name and credit card number
    - A product is represented by a product code and price
    - There is a one-to-many relationship between a customer and products
  - **From:**
    - “Principles of Database and Knowledge-Base Systems”, J.D. Ullman, Volume I, Computer Science Press, 1988, p. 32.



# Application Data Models

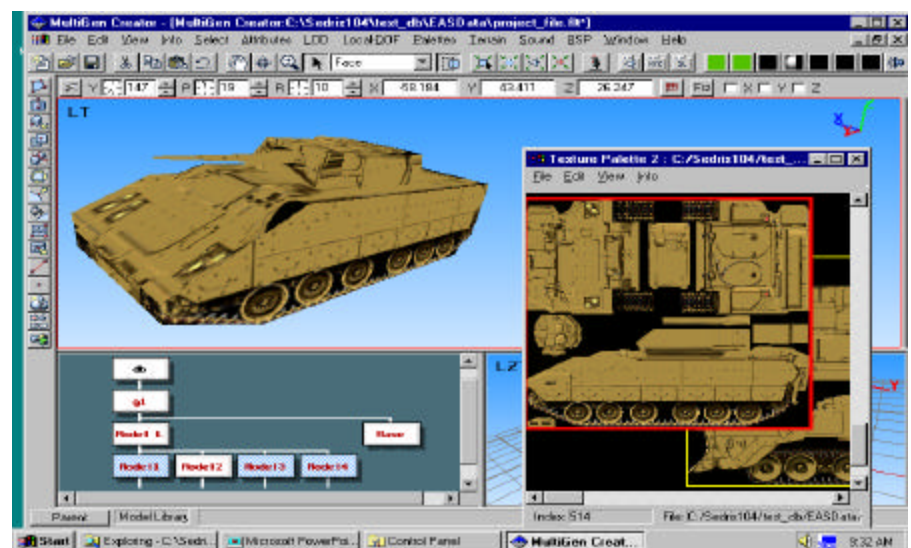
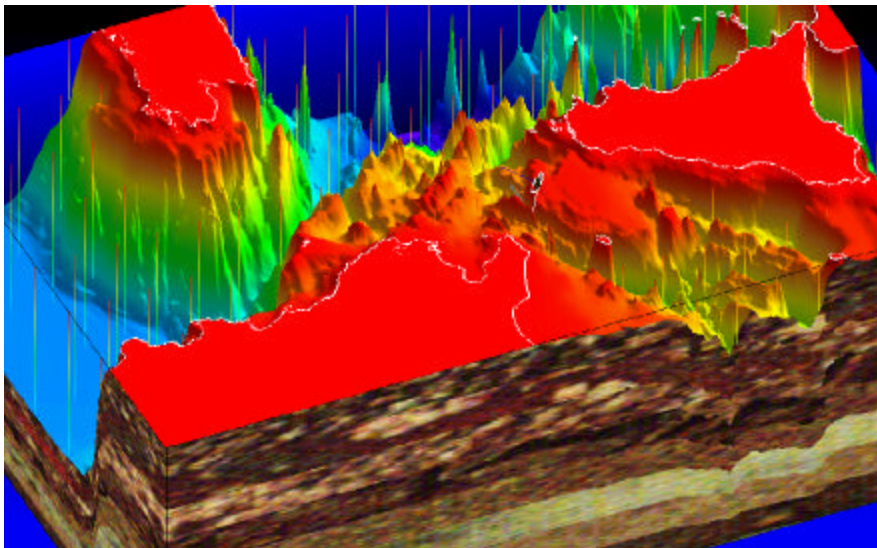
---

- **Every environmental application has a “native” data model.**
  - Native to the domain or application scope
  - Optimized for the application
  - Limited to only the data elements used within its abstraction of the environment
  - Has a specific representation of environmental elements
- **Application data models are derived based on the purpose of the application such as:**
  - To drive down the road
  - To simulate an aircraft
  - To find closest fishing spot
  - To design a vehicle





## Application Data Models Views





## Same Object - Differing Views

---

- **What we think of an object depends on our use, interaction, and perception of that object ... our concept**
- **For example, a bridge is**
  - an obstacle to sailboats with long masts
  - a connector in road networks
  - a target during warfare
  - a load-bearing structure to an engineer
- **These differing views are also reflected in the way applications process environmental data about the same object**
- **SEDRIIS is designed to support multiple independent or integrated views**



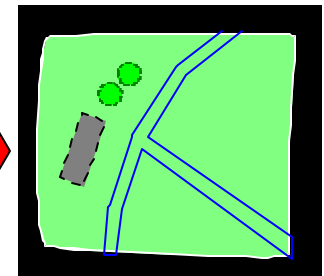


# Different Representations of the Same Environment

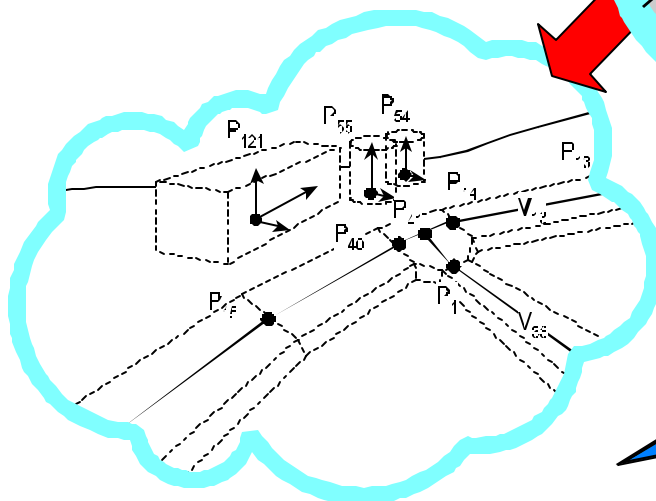
Visual Database



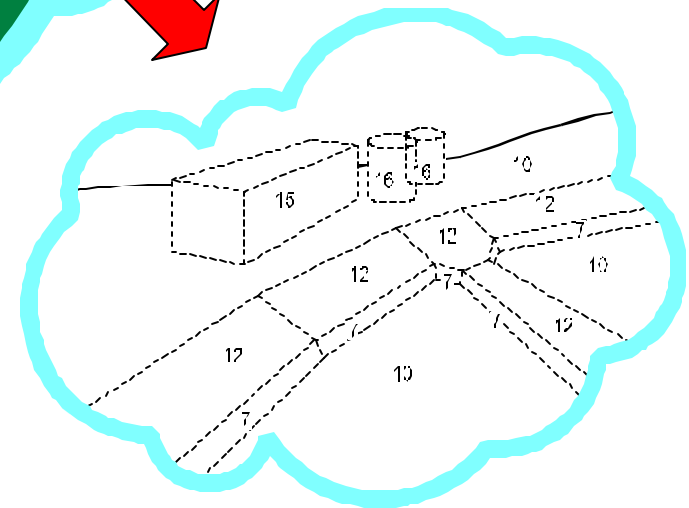
Electronic Maps



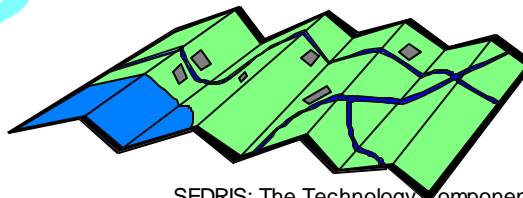
Mobility Database



CGF Database



Paper Maps





# A DRM for the Environment

---

- **How do you provide a mechanism for all environmental data; i.e., take into account all native data models?**
- **One approach: Capture all elements into one Data Model**
  - Hard to agree on
  - Hard to determine all possible applications or combinations
  - Impossible to capture all possible data models (never ending task)
- **The SEDRI DRM approach:**
  - Decompose known environmental data models into components
  - Generalize and extrapolate variations
  - Unify common components
  - Verify completeness by forward and reverse mappings to known data models
  - Use the DRM as the building blocks for common representation
  - Plan for change



# From DMs of the Environment ... to a DRM *for* the Environment



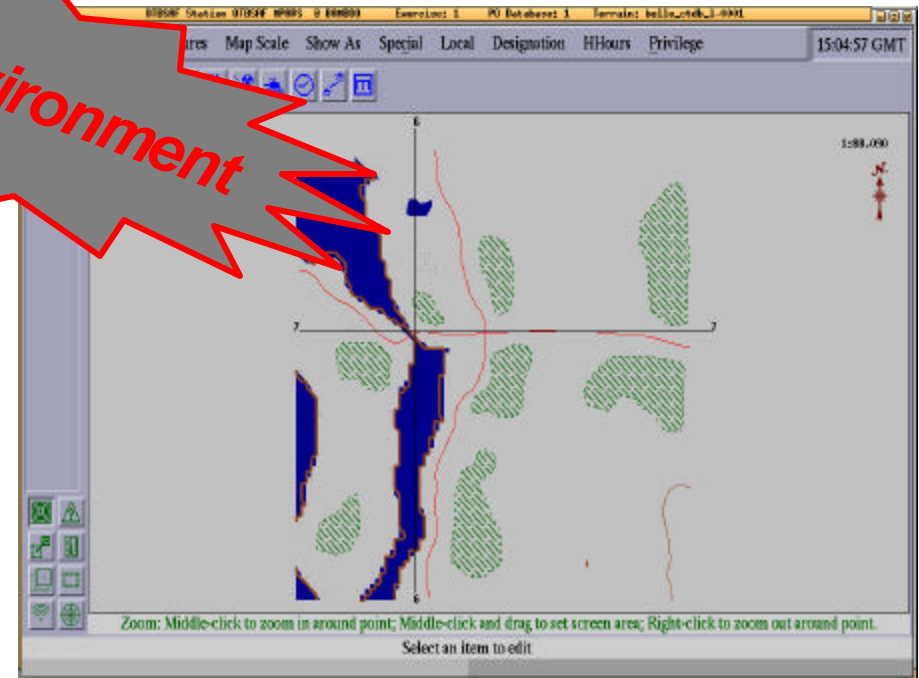
**Same Environment**

- **Common DM Based Approach**

- Different native data models
- Difficult to map between them
- “Value added” data needed for interchange is external to both models

- **SEDRIS DRM Based Approach:**

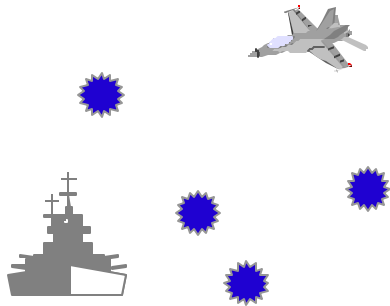
- One DRM supports representations of both views
- Unambiguous representation aids in mapping between a native DM and the DRM
- “Value added” data may be fully integrated with both views



# **Spatial Reference Model (SRM)**

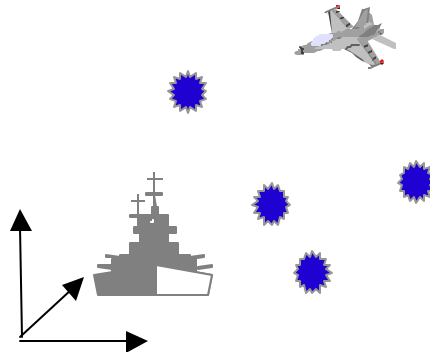


# Representation begins with Location ...



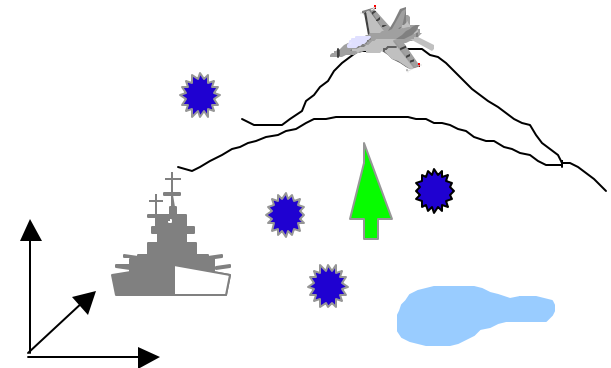
## Systems

The void ...



## Systems, *where?*

Start with locating your systems; sometimes that's about all you could afford in legacy simulations.



## Systems, *and what else?*

Define the context within which systems engage; and that context can advantage, or disadvantage, ...

***Defining and using a consistent spatial reference framework is critical for interoperability***

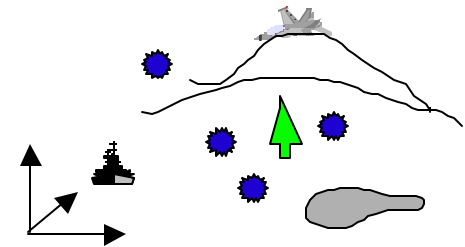
- System models (men, material, ...)
- Environmental data, models, phenomena



## ... but Location is Not Enough

- The key in interoperability is not so much where you are, but *what you can interact with ...*
  - Remembering that “what” includes both systems and components of the environment itself
- A complete SRM must address:
  - Direction (azimuth and elevation angle)
  - Range
  - (support for) Geometric intervisibility

- *Location interconversions between common spatial reference frames do not necessarily preserve these*



*Systems, and what else?*

The environmental representation continues with defining the context within which systems *interact*; and that context can advantage, or disadvantage, them ...



# SRM Requirements

- **Completeness, must:**
  - include coordinate systems in wide usage.
  - tie those systems together into a common framework.
  - educate the system developer.
    - E.g., What's a ORM, a vertical offset surface, a geoid?
- **Accuracy**
  - Generally higher than required for C4ISR systems.
  - E.g., typically better than 1 cm. up past geosynchronous orbit (nominally 1 mm near the ERM surface).
  - Computation of operations in spatial frames must not impact application validity.
- **Performance**
  - Never fast enough!
  - Many environmental data sets dominated by location data
  - Federate costs for distributed systems using heterogeneous coordinate systems can be substantial (e.g., 20% or more).

***SEDRI algorithms and implementations are very accurate and 2-10x faster than algorithms/implementations reported in the literature.***



# Spatial Reference Frames

***Spatial Reference Frames (SRF)*** serve to locate coordinates in a multi-dimensional space (generally either two- or three-dimensional). Specified in two parts:

A geometric description (model) of a reference object embedded in (and serving to orient) that frame – referred to as an ***Object Reference Model (ORM)*** an Earth Reference Model (ERM) is a special case of an ORM.

A ***Coordinate System (CS)*** specifying how a tuple of values uniquely determine a location with respect to the origin of that frame. By extension, that tuple also specifies a location with respect to the reference object.

$$SRF = ORM + CS$$

**There are no “naked” coordinate systems**





# SRM Summary

---

- **Unambiguously specifying location**
  - Common SRF provided in implementation
  - Used within the DRM to store location information
  - Used within the DRM to store SRF information
- **Efficient conversion of location data**
  - Platform and language independent implementations to change from one SRF to another
  - Provides a framework to inter-convert between standard SRFs as well as new SRFs

SRM



# SRM Tutorial

---

## **Spatial Reference Model (SRM)**

**1:30 PM Tuesday, 6 January**

**Room: Sapphire**

### **WHO SHOULD ATTEND:**

Those interested in gaining a more complete understanding of the SRM and the theory of accurately representing spatial locations in modeling

# Environmental Data Coding Specification (EDCS)



# Environmental Data Coding Specification

---

## *Unifies characterizations of environmental “things”*

Regardless of how represented:

Feature or Geometry or Data Table or Model or ...

Whether individual primitives or structured collections of primitives:

Furniture vs. Room vs. Building vs. Facility vs. Region

## *Separates enumerations from Data Representation Models*

Evolve at different rates for different reasons

It's a big world to capture ...

## *Answers three types of questions:*

*What is it?* Classifications and Features

*What are its additional clarifying characteristics?* Attributes and Values

*What are its characteristic measures?* Units of Measure and Scale



# Classifications (and Features)

## 1. What is it?

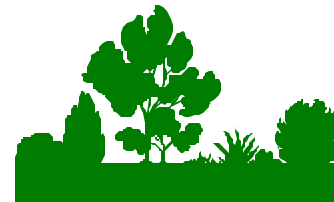
building, river/stream, air warning light, ocean floor



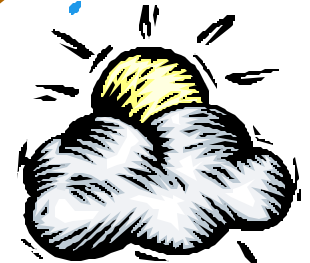
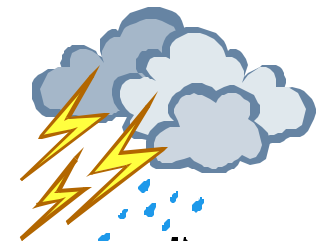
Animal?



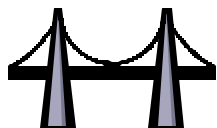
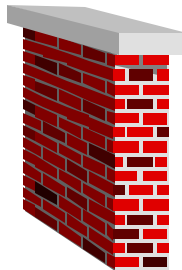
Water?



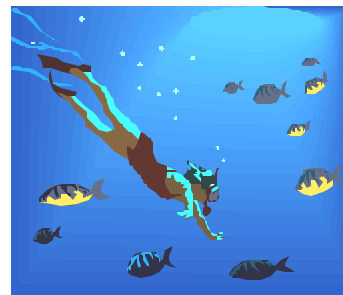
Vegetable?



Weather?



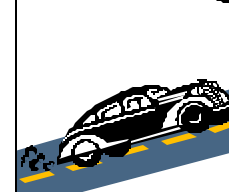
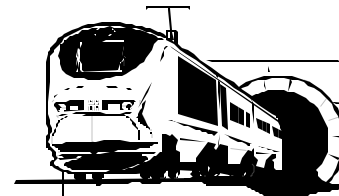
Structure?



Vehicle?



Mineral?



Celestial?



# Attributes (and Values)

## 1. *What is it?*

building, river/stream, air warning light, ocean floor

## 2. *Additional clarifying characteristics?*

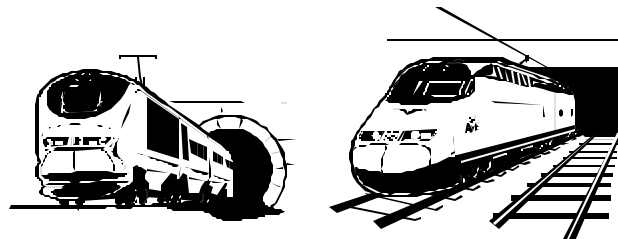
lighthouse, 1.5, red, coral



Vegetation Type?



Building Function?



Overhead Clearance?





# Units of Measure and Scales

## 1. *What is it?*

building, river/stream, air warning light, ocean floor

## 2. *Additional clarifying characteristics?*

lighthouse, 1.5, red, coral

## 3. *What are its characteristic measures and scales?*

kelvin, decametre, kilometre/hour; micro, tera, deci



**How fast?**  
kilometres per hour




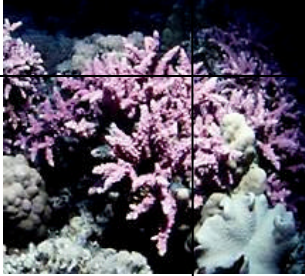
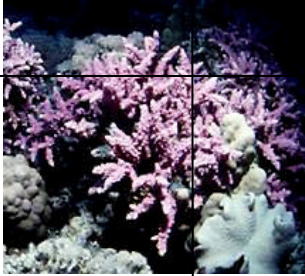
**How warm?**  
kilokelvin



**How tall?**  
decametres



# Putting EDCS Entries Together Clarify the Object's Description

<i>What is it?</i>	<i>How is the object characterized ?</i>	<i>How is the object measured?</i>
Classifications	Attributes	Units of Measure & Scales
Building	<p>With the <u>function</u> of a <i>Lighthouse</i></p> <p>Whose <u>height</u></p> 	<p>is 3.05 <u>decametres</u></p>
River Stream	<p>Whose <u>speed</u></p> 	<p>is 1.5 <u>metres per second</u></p>
Ocean Floor	<p>Which is <u>composed</u> of <i>Coral</i></p> <p>Whose <u>density</u></p> 	<p>is 0.97 <u>kilograms per cubic decimetre</u></p> <p><a href="http://physics.nist.gov/cuu/Units/index.html">http://physics.nist.gov/cuu/Units/index.html</a></p>





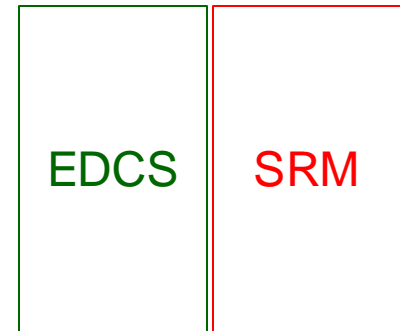
# EDCS Basics

- **The EDCS provides mechanisms to unambiguously specify objects used to model environmental concepts.**
  - The EDCS supports the encoding and communication of qualitative and quantitative information associated with natural and artificial environments
- **The EDCS specifies a collection of nine dictionaries of environmental concepts, as well as:**
  - Guidelines for expanding these dictionaries through registration,
  - Conventions for applying the encodings in information processing applications,
  - A functional interface to convert between numeric values given in different units of measure and scales, and
  - Organizational Schema and Groups to aid in searching specific dictionaries
- **An EDCS Dictionary Entry for a given environmental concept minimally includes:**
  - A Definition which is a precise statement of the nature, properties, scope or essential qualities of a concept embodied in the entry
  - A Label which is a compact and human-readable designator that is used to denote a concept; this is represented as a character string
  - A Code which is a compact, and not necessarily human-readable, designator that is used to denote a concept; this is represented as an integer
  - Each entry is unique and defines one concept
- **EDCS is not a Data Model or a Taxonomy**
  - Allows a definition to reference other concepts in order to make best definition



# EDCS Summary

- **Answers**
  - *What is it?*
  - *What are its additional clarifying characteristics?*
  - *What are its characteristic measures?*
- **Provides a data dictionary**
- **Expandable**
- **Separates representation from semantics**





# Environmental Data Coding Specification

**8:30 AM Wednesday, 7 January**

**Room: Sapphire**

### **WHO SHOULD ATTEND:**

Those desiring to define the semantics of environmental data (the environmental "things" and what they "mean"), either as data providers, data consumers, or both. Both project managers and technical implementers will benefit from this tutorial.

# **Data Representation Model (DRM)**



## DRM Basics

---

- **The DRM is a set of classes and the data types used to specify their data elements**
- **The DRM defines formal relationships between these classes**
- **The DRM specifies a set constraints or requirements on instances of classes**
- **Actual data sets contain object instances of DRM classes**



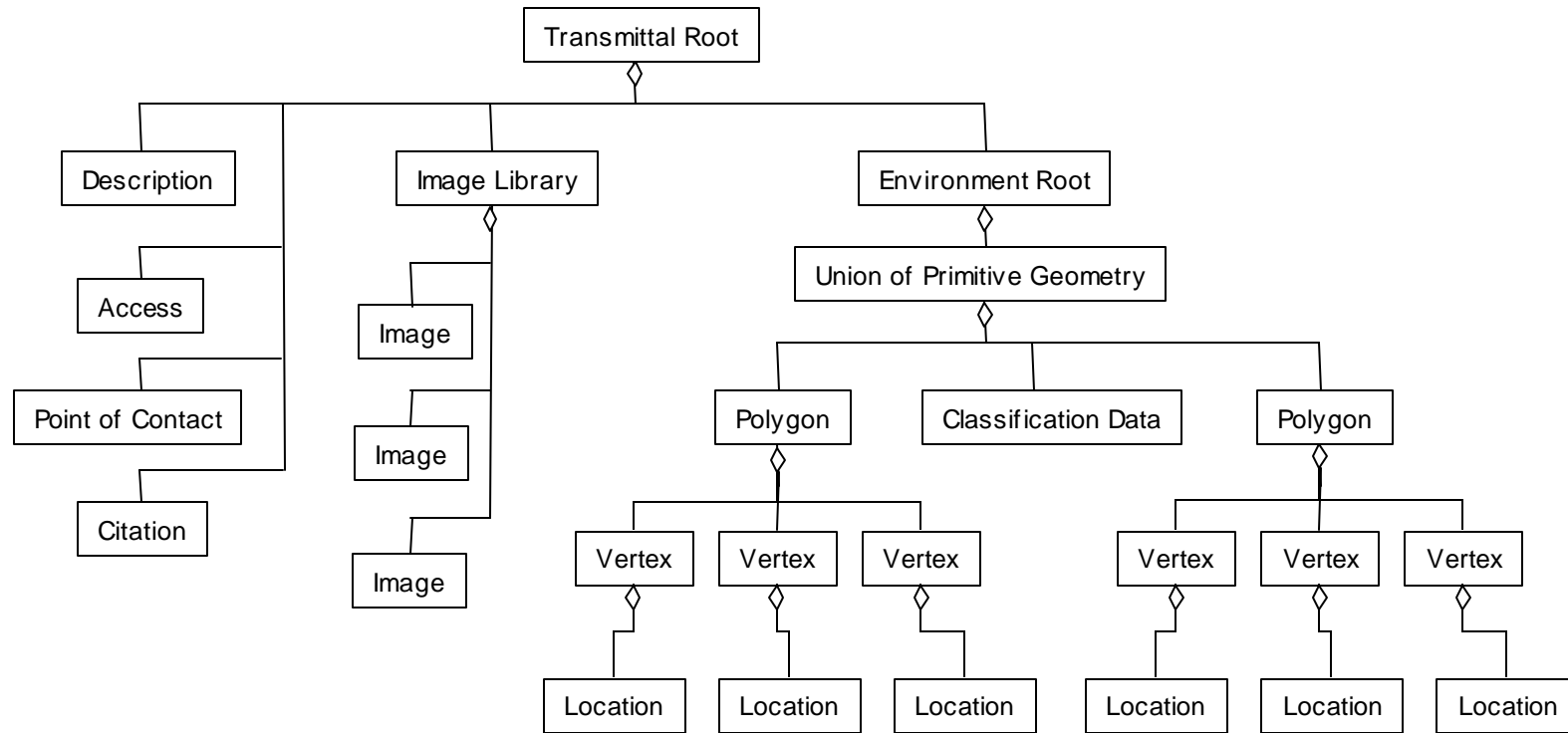
## DRM Basics

---

- The formal relationships between classes specify what relationships are allowed to exist between instances of those classes and what those relationships mean
- The constraints further refine requirements for objects in specific contexts
- A SEDRIIS data set or database is called a *transmittal*
- The objects contained within a transmittal are organized as a tree in terms of aggregate/ component (whole vs. part) relationships between objects.



# DRM Basics



- A sample transmittal provided as a DRM class instance



# An Example of a DRM Class Definition

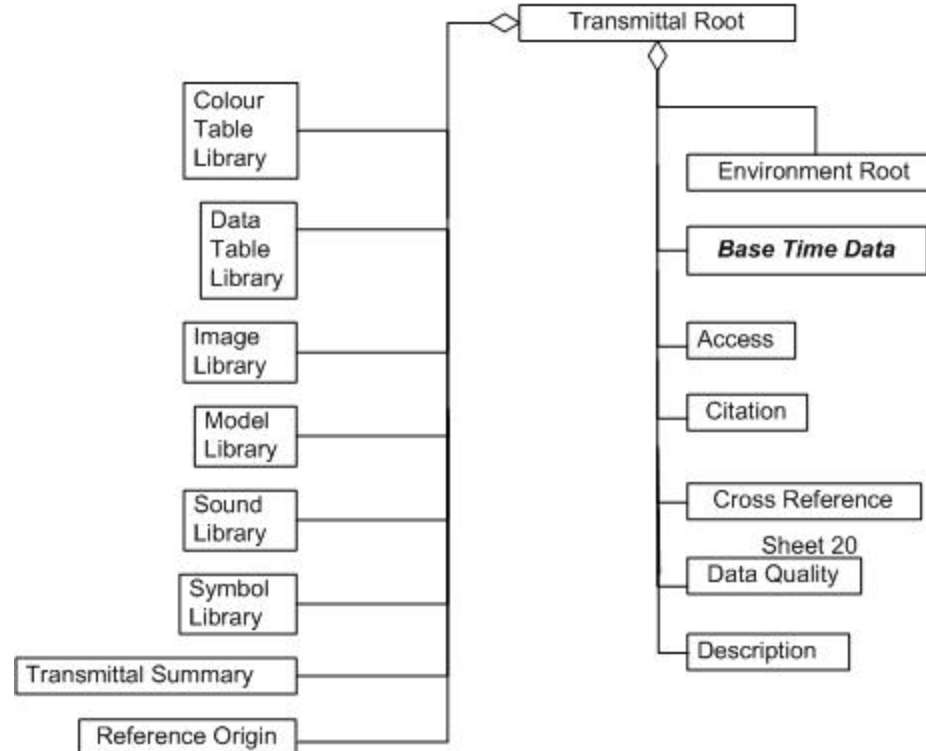
<b>Class</b>	<Transmittal Root>	
<b>Super Class</b>	SEDRIIS Abstract Base	
<b>Sub Class</b>	None	
<b>Description</b>	The hierarchical root of all objects in a single SEDRIIS transmittal.	
<b>Field Elements</b>	name, major_DRM_version, minor_DRM_version, interim_DRM_version, major_EDCS_version, minor_EDCS_version, interim_EDCS_version <sup>2</sup>	
<b>Possible Components</b>	<div> <div> &lt;Base Time Data&gt;  &lt;Data Table Library&gt;  &lt;Image Library&gt;  &lt;Property Set Table Library&gt;  &lt;Sound Library&gt;  &lt;Transmittal Summary&gt;  &lt;Citation&gt;  &lt;Data Quality&gt; </div> <div> &lt;Colour Table Library&gt;  &lt;Environment Root&gt;  &lt;Model Library&gt;  &lt;Reference Origin&gt;  &lt;Symbol Library&gt;  &lt;Access&gt;  &lt;Cross Reference&gt;  &lt;Description&gt; </div> </div>	
<b>Constraint</b>	Objects of this class cannot be published.	





# DRM Class Diagram Specification

- Similar class definitions exist for all 303 DRM classes
  - Provided within the DRM documentation in HTML
- UML diagram providing the same basic information:





# Abstract vs. Concrete DRM Classes

---

- Abstract DRM classes may not be instantiated as objects.
- Concrete DRM classes may be instantiated as objects.
- UML Notation:

Concrete Class

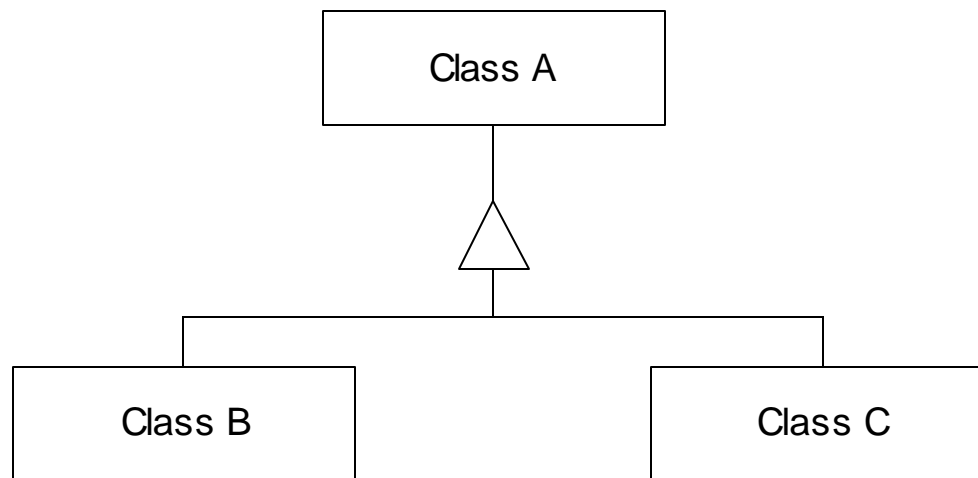
*Abstract Class*

- Abstract classes denoted by *italics*



# Inheritance Relationship

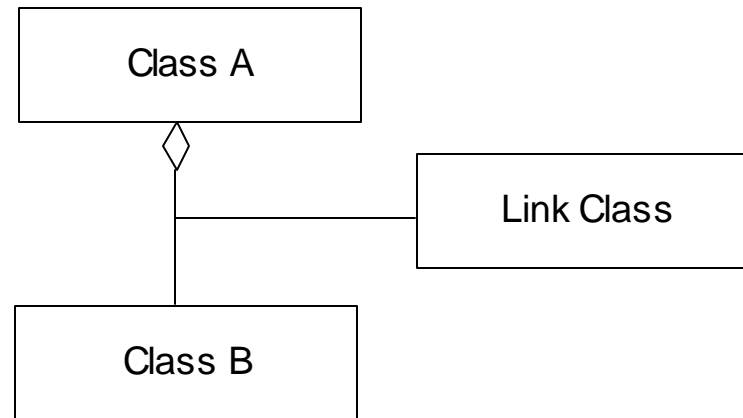
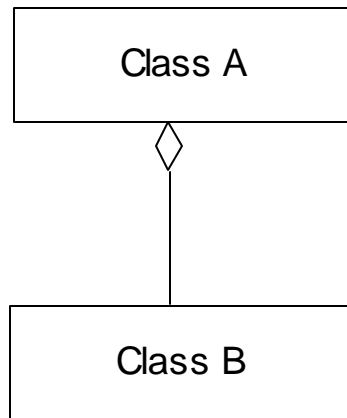
- Defines a class as being the super class of another class, also referred to as the “Is-A” relationship.
- UML Notation:





# Aggregation Relationship

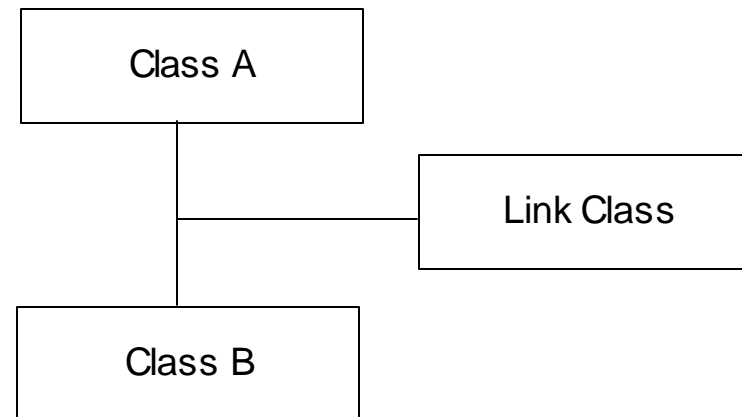
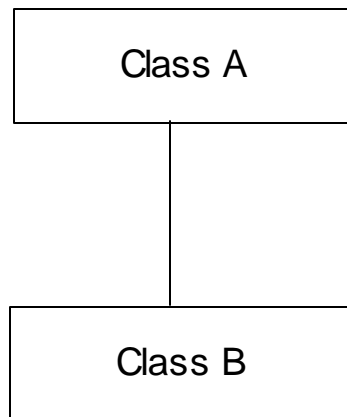
- A class contains another class, also referred to the “Has-a” relationship.
- Some aggregations have a link class to describe the relationship further.
- UML Notation:





# Association Relationship

- A relationship exists between two classes other than an aggregation relationship.
- Some associations have a link class to describe the relationship further.
- UML Notation:





## Relationship Multiplicity

- The DRM defines how many objects of the DRM classes can be related to each other.
- Allowed values:

_____	<input type="text"/>	Exactly one
0 .. 1	<input type="text"/>	Zero or one
*	<input type="text"/>	Zero or more
N .. *	<input type="text"/>	N or more
{ordered}	<input type="text"/>	Ordered



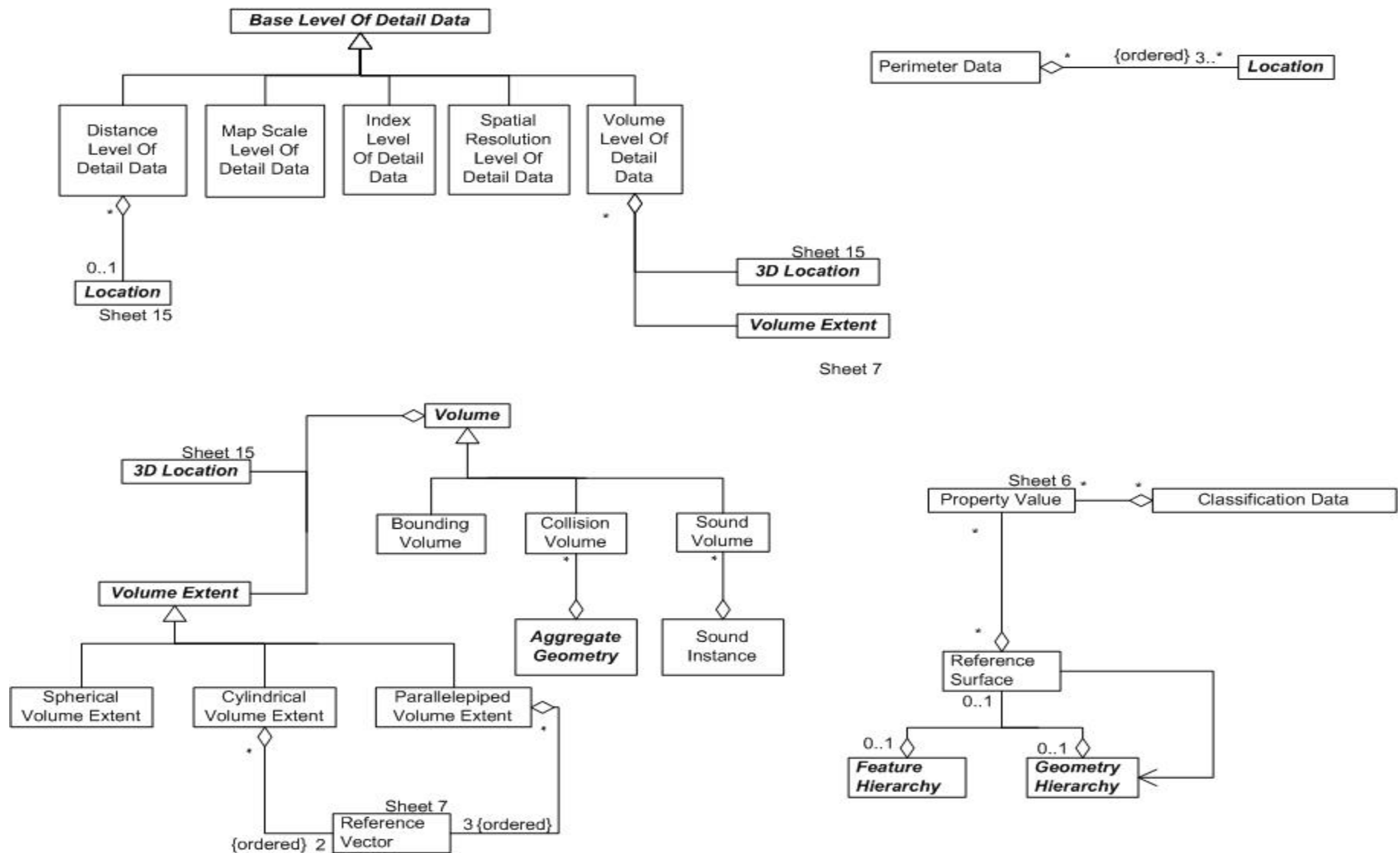
# DRM Diagram Reading Exercise

---

- Since we are now familiar with the basics of reading DRM diagrams, we have a simple exercise
- The next two slides contain two DRM diagrams, which we will review
- The subsequent four slides contain various instance diagrams that we will evaluate for DRM conformance
- For each instance diagram:
  - Determine if it is conformant to the DRM Diagrams provided
  - Answer “Yes”, “No”, or “Indeterminate”
    - If the answer is “No”, then correct the instance diagrams
    - If the answer is “Indeterminate” then determine what additional information is required?



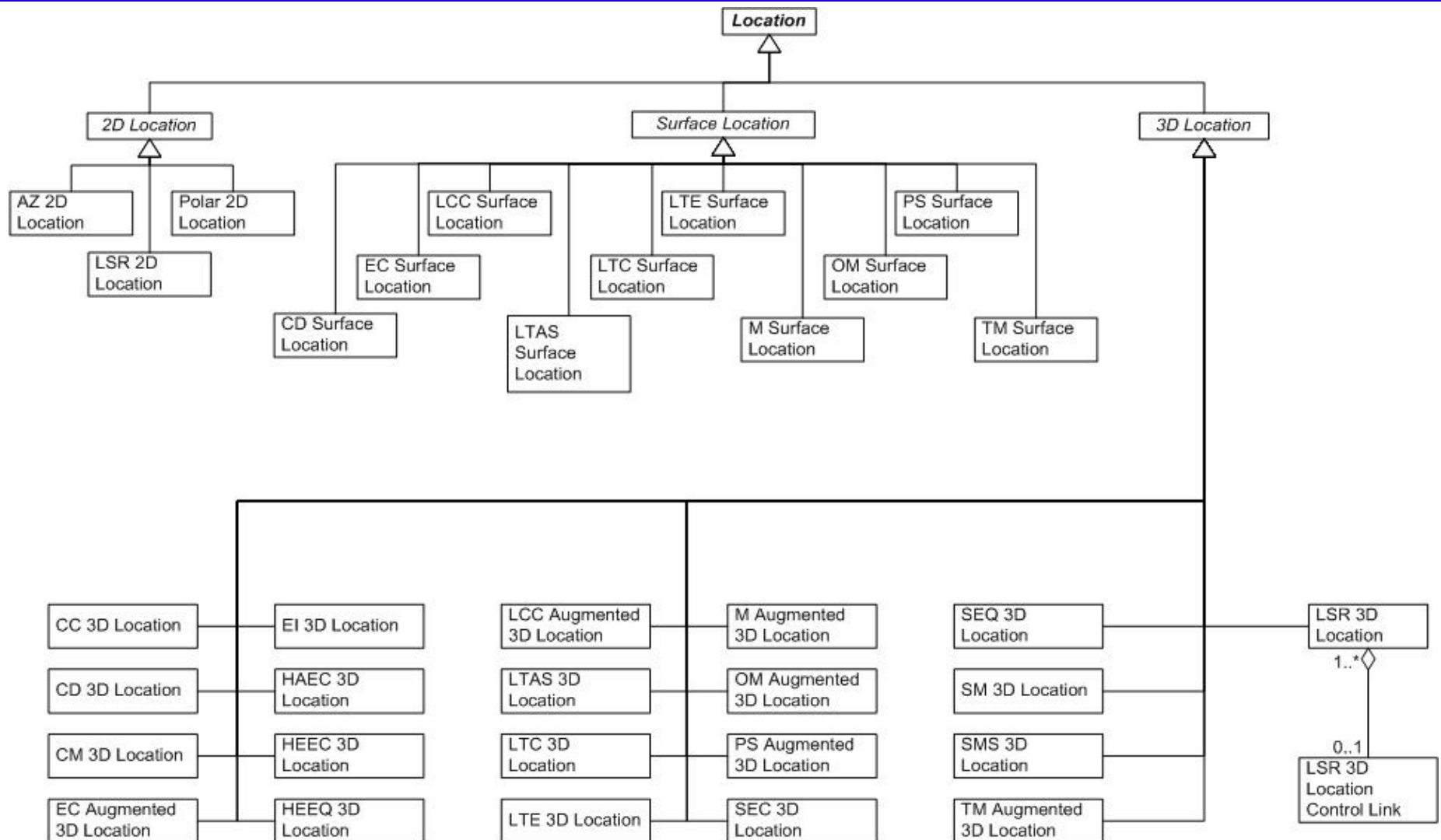
# DRM Diagram Reading Exercise







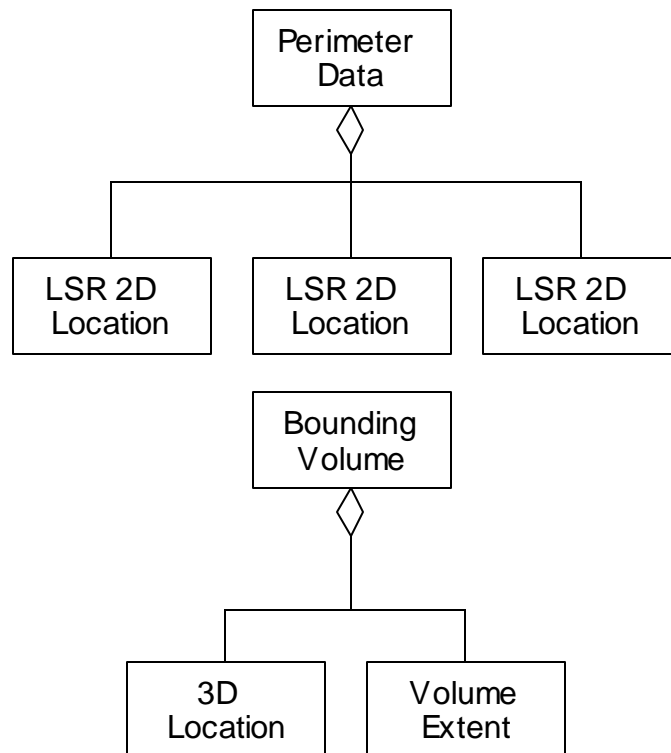
# DRM Diagram Reading Exercise





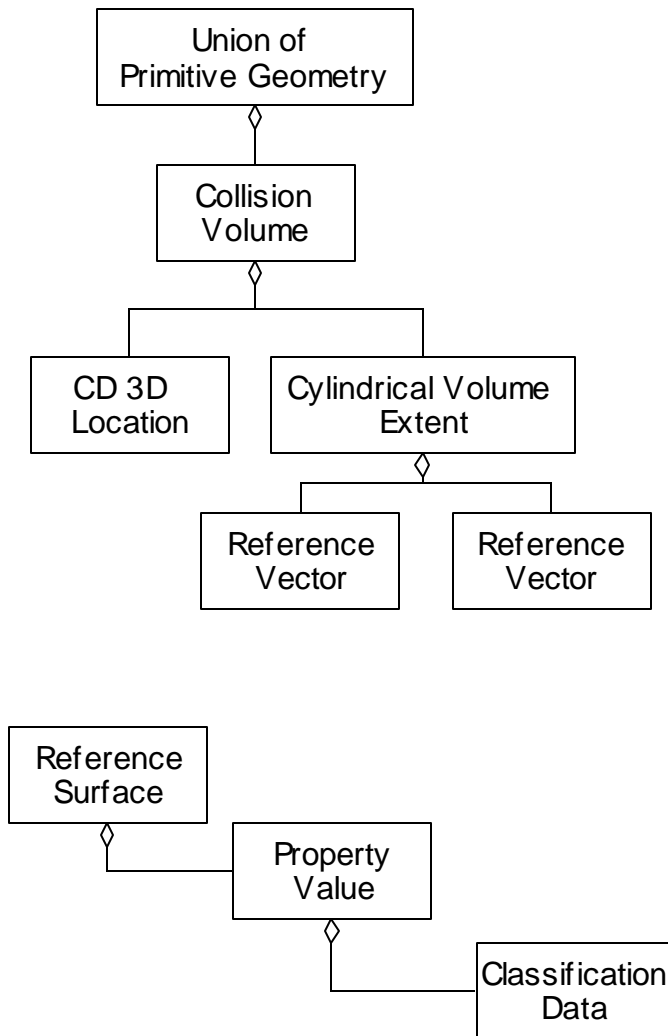
# DRM Diagram Reading Exercise

- Are the following valid DRM object instance diagrams?
  - Yes, No, Not Enough Information to Determine
  - If not fix



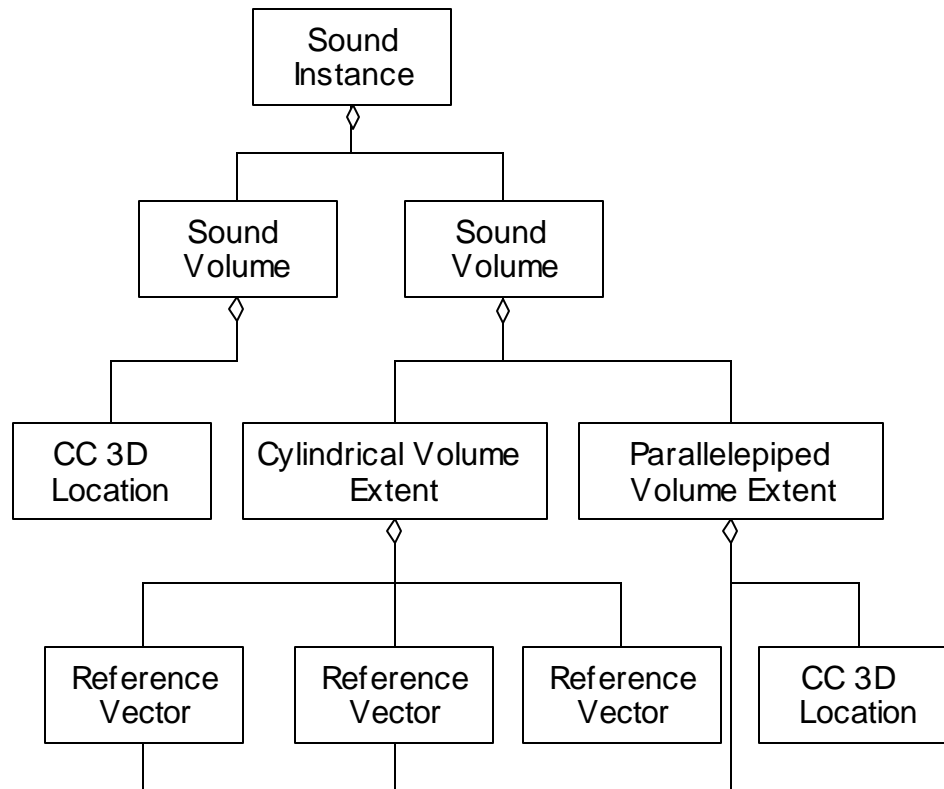


# DRM Diagram Reading Exercise





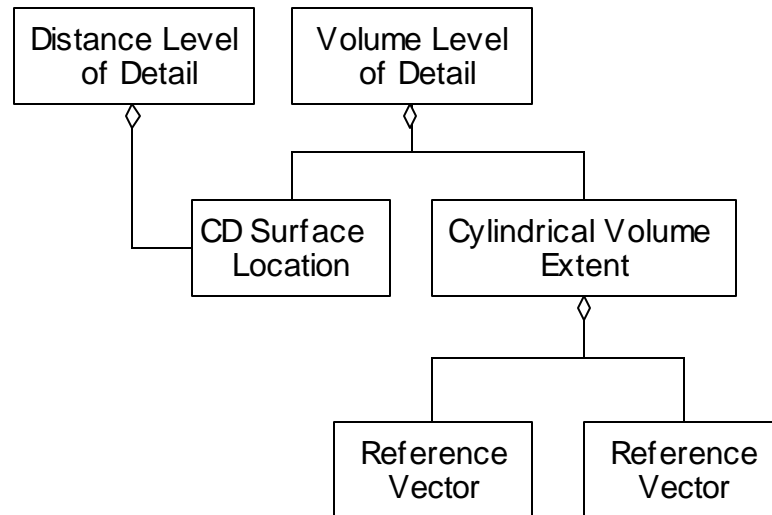
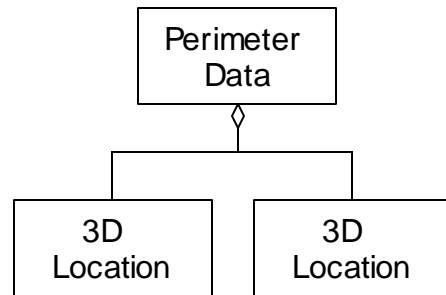
# DRM Diagram Reading Exercise





# DRM Diagram Reading Exercise

---





# The DRM is Composed of ...

- **Organizer and container classes**
  - Classes used to organize other classes
  - Examples
    - <Transmittal Root>, <Library>, <Environment Root>, <Feature Hierarchy>, <Geometry Hierarchy>
- **Primitive representation element classes**
  - Geometry classes
    - Physical/surface representations of real world objects (is: 3D polygons, images, lighting, etc.)
  - Feature classes
    - Higher level abstraction of real world objects (ie: area, line, and point representations)
  - Topology
    - Concise, mathematical definition of inter-Feature or inter-Geometry object relationships
- **Properties**
  - Classes which describe the representation elements in a transmittal
  - Examples
    - <Classification Data>, <Property Value>, <Colour>, & metadata classes



## **<Transmittal Root>**

---

- **Every transmittal contains one <Transmittal Root> object**
- **Every object in a transmittal is ultimately aggregated by the <Transmittal Root> object**
- **A <Transmittal Root> organizes**
  - **<Environment Root> instances that represent some environment**
  - **<Library> objects that serve as repositories of representations of environmental objects that tend to be shared and reused**
  - **metadata that applies to the entire transmittal**



## <Library>

- The <Library> classes exist to organize a common representation that is multiply referenced within a transmittal.
- Common representations are found within the <Library> objects and are instantiated under <Environment Root> objects
- Typical use cases
  - An object exists in the domain being represented that isn't a fixed part of the environment. For example: a vehicle, a person, or an animal
  - A 'generic' representation exists for which lots of copies will be made. For example: a house, a tree, a streetlight





## Referencing the <Library>

- Each <Library> subclass has classes for referencing their contents:

- <Colour Table Library>

- <Data Table Library>

- <Image Library>

- <Model Library>

- <Feature Model Instance>

- <Property Set Table Library>

- <Sound Library>

- <Symbol Library>

- <Colour Index>

- <Property Table Reference>

- <Image Mapping Function>

- <Geometry Model Instance>,

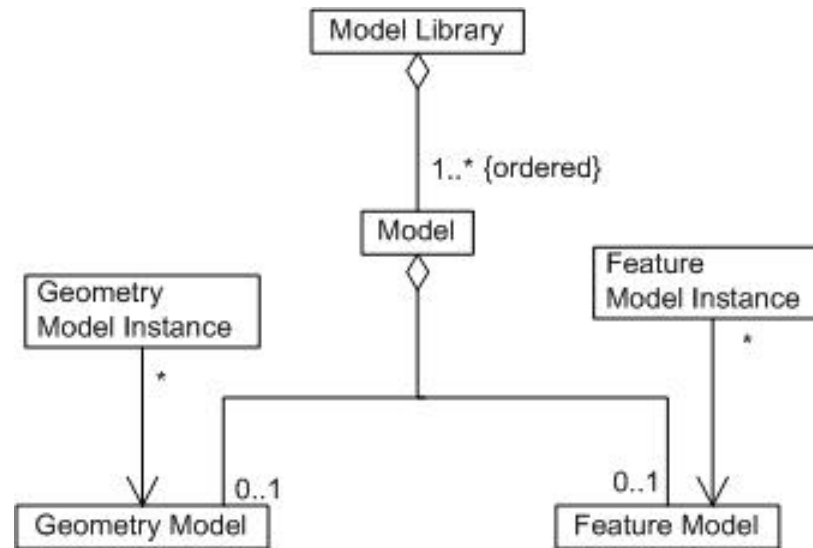
- <Property Set Index>

- <Sound Instance>

- <Label>



## <Library> Referencing Example



- A <Model Library> contains an ordered set of <Models> with a <Feature> representation or a <Geometry> representation, or both.
- The model is 'instanced' under an <Environment Root> object by either a <Geometry Model Instance> object or a <Feature Model Instance> object
  - Done by an association from the instance to the model.



# Organizing <Feature> and <Geometry> primitives

---

- **DRM Classes are defined to organize primitives by providing semantic information to better organize the environmental data**
- **For <Geometry> primitives:**
  - **The organizing principles correspond to the subclasses <Aggregate Geometry>**
  - **All primitives ultimately reside under a <Union Of Primitive Geometry> or <Property Grid Hook Point> object**
- **For <Feature> primitives:**
  - **The organizing principles correspond to the subclasses of <Aggregate Feature>**
  - **All primitives ultimately reside under a <Union Of Features> object**



# Organizing Principles

---

- **Union**
  - Organize primitives that need to be grouped together but for which no other organizer is more suitable.
  - A “bag of” points, polygons, etc
- **Alternate Hierarchy**
  - To provide two or more representations of the same underlying data
  - Organized as separate branches of hierarchy tree
  - The <Hierarchy Data> associated with each branch describes that particular representation
- **Spatial Index**
  - Organizes data into a spatial grid
  - Each branch of the hierarchy contains one grid



# Organizing Principles

---

- **Quadrant**
  - Divides the representation into four spatial regions
  - Each branch contains one of the quadrants
- **Octant**
  - Divides the representation into eight volumes
  - Each branch contains one of the octants
- **Perimeter**
  - Organizes data into irregularly shaped partitions
  - Each branch of the hierarchy specifies the perimeter of the region
  - Each branch contains primitives that fall within the specified perimeter



# Organizing Principles

---

- **Classification**
  - Organizes data according to its EDCS Classification.
- **Level Of Detail**
  - Organizes different representations of the same underlying data that differ in level of detail.
  - Each branch contains one of the octants
- **Continuous Level Of Detail**
  - Is specific to primitive geometric representations, such as polygonal representations
  - Provides a mechanism for specifying an arbitrary number of levels of finer and finer detail
  - Allows the consumer to decide under what conditions the end application should switch from a coarse representation to a more detailed one, or vice versa.



# Organizing Principles

---

- **Separating Plane & Animation**
  - Specific to geometric primitives
  - Organize by planes that partition the geometry into volumes to assist the rendering process.
- **Animation**
  - Specific to geometric primitives
  - Organize by creating an animation sequences.
- **Time**
  - Organizes data temporally
  - Each branch specifies a different time period
- **State**
  - Organizes based on the state being represented



# The Primitive's Classes

---

- **Features**
  - <Point Features>, <Linear Features>, <Areal Features>
- **Geometry**
  - <Camera Points>
  - <Points>, <Lines>, <Arcs>, <Ellipses>, <Elliptic Cylinders>
  - <Polygons>, <Finite Element Meshes>
  - <Light Sources>
  - <Sounds>
  - <Property Tables>, <Property Grids>
  - <Images>
- **Model Instances**





# Properties in the DRM

---

- The DRM defines a large set of classes used to provide the properties of primitives
- The object instances of these classes are added as components of the primitive objects
- Location is the first property:
  - *Coordinates* are pairs (2D) or triples (3D) of real numbers that designate the position of a point in a coordinate system.
  - A *coordinate system* is a set of rules by which a coordinate can spatially relate a location to a unique coordinate system origin and associated axes.
  - A *spatial reference frame* ties a coordinate system's origin to some Object Reference Model, such as a model of the Earth, so that it is no longer arbitrary but tied to the real world.



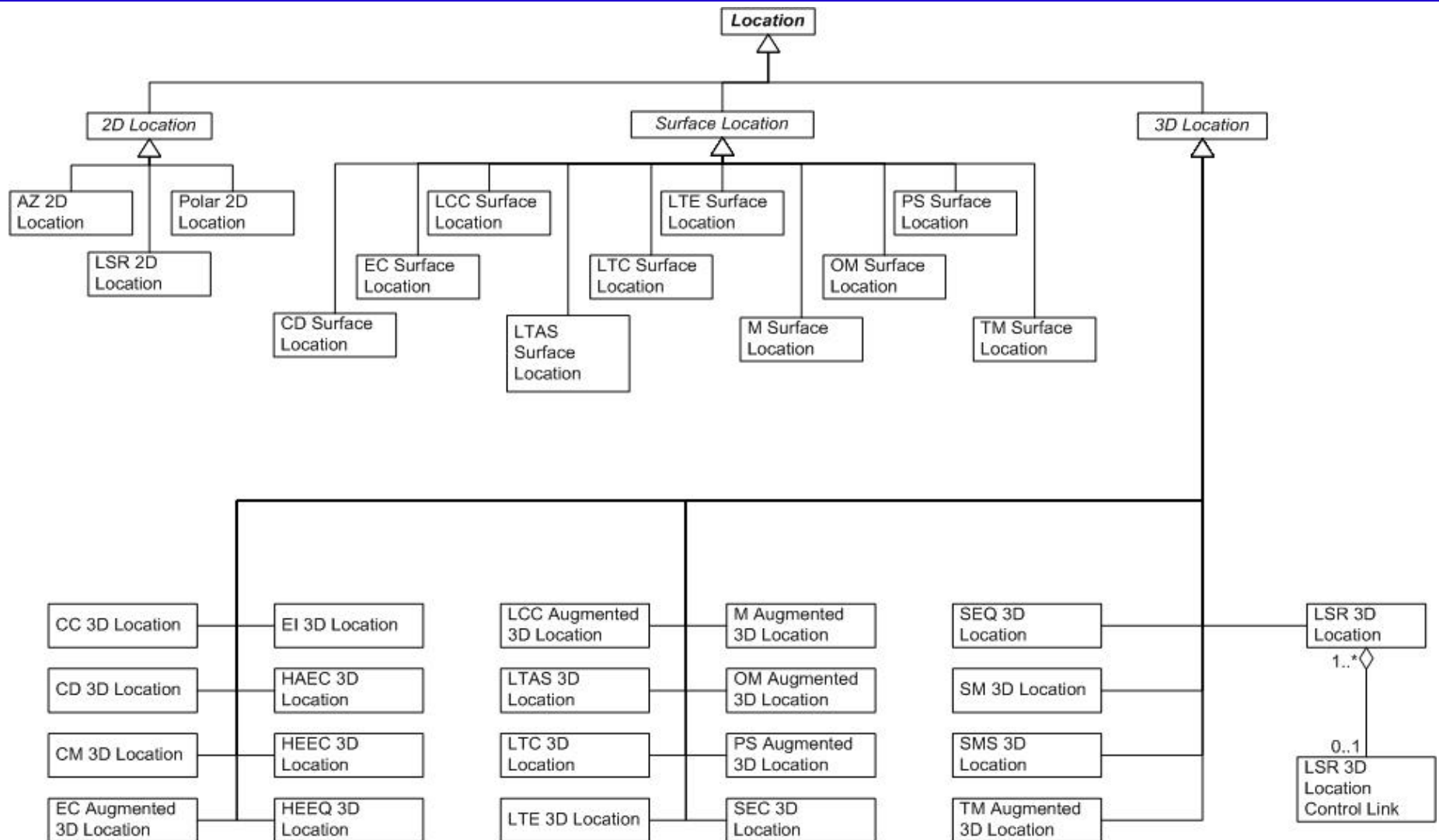
## Coordinates in the DRM

---

- In the DRM, coordinates are represented as <Location> classes
- Each <Location> belongs to a specific SRF
- The SRF information is provided at a few organizer classes, and not at the individual <Location> classes
- <Environment Root> is one of the 5 classes of objects that specify SRF information
- The DRM requires that a <Location> must be valid in the SRF it appears in.



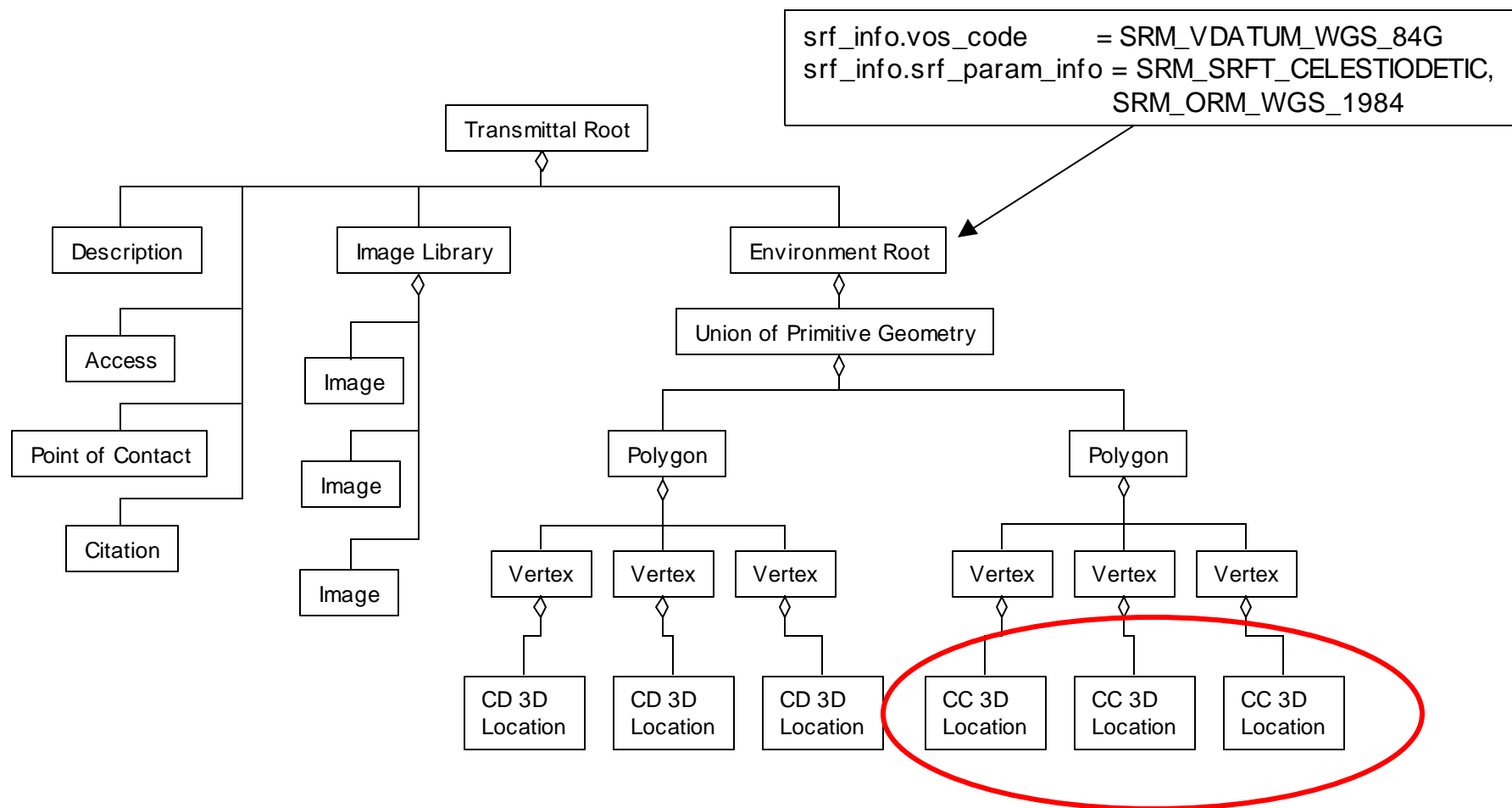
# <Location> classes





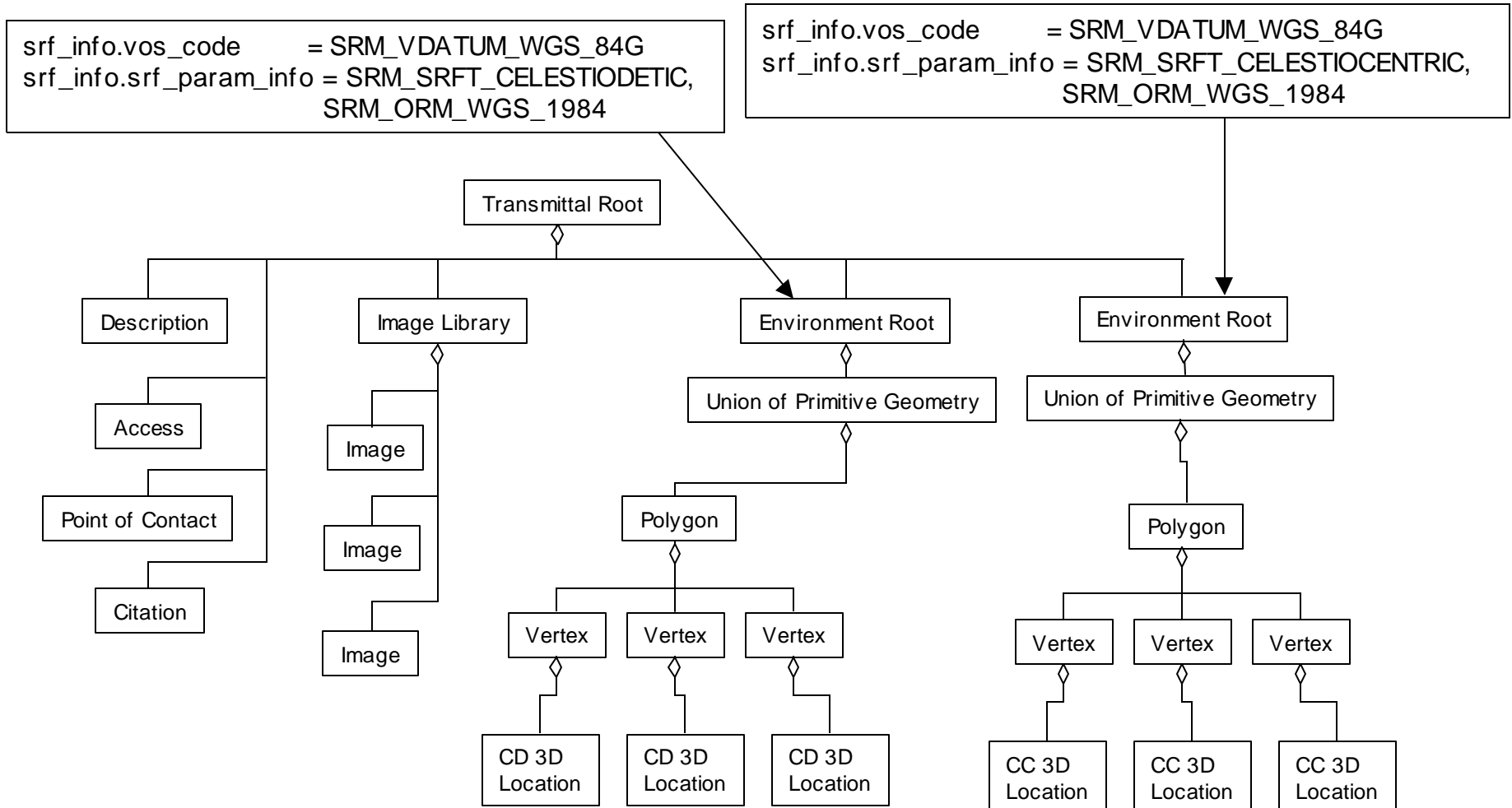


# Incorrect Location





# Incorrect Location - Corrected





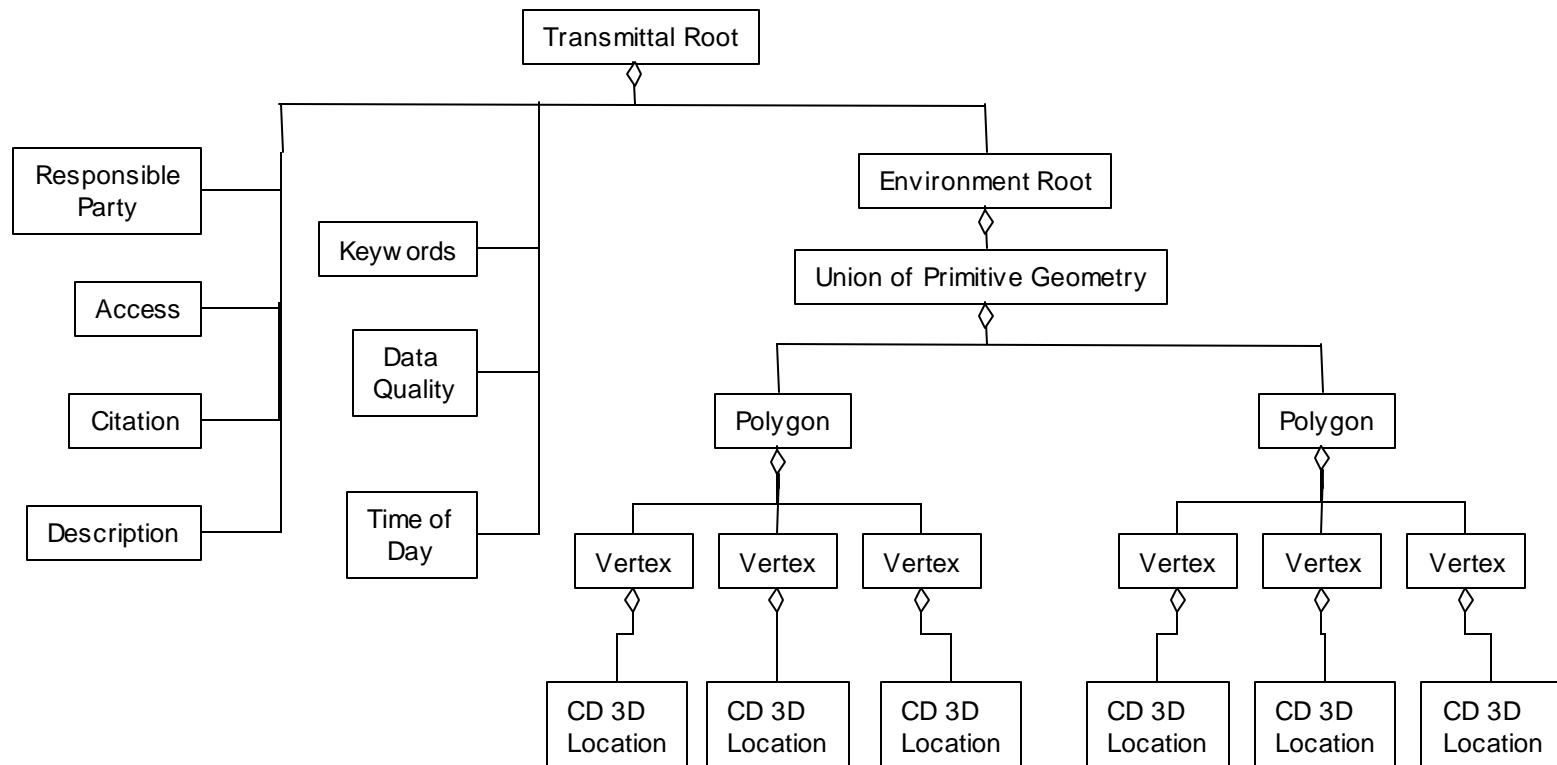
## Semantics in the DRM

---

- **DRM classes are used to build a representation of “real world” objects**
- **What are real world objects?**
  - You are
  - This building
  - The city we are in
  - The country we reside in
  - The transportation we used to get here
- **So, anything we can see or touch is a real world object**
- **Commonly referred to as “features”**
- **This can be extended to things we can model such as a light saber or a hobbit**



# What does this transmittal represent?



- Which SEDRIS component is designed to answer this question? \_\_\_\_\_





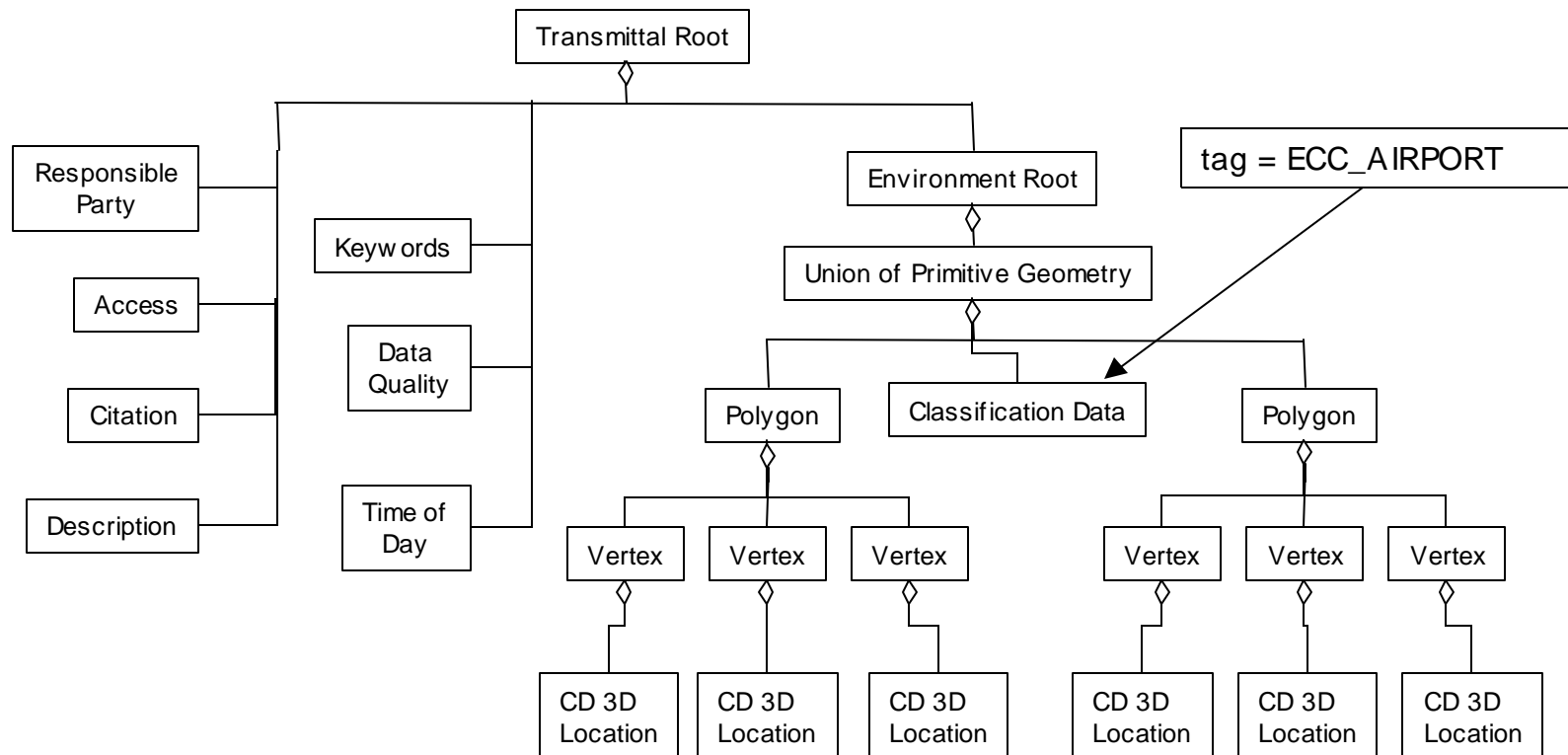
## **<Classification Data>**

---

- **An EDCS Classification Code provides the semantic meaning in the DRM class <Classification Data>**
- **A <Classification Data> instance attached to an object specifies what the object represents in the world**



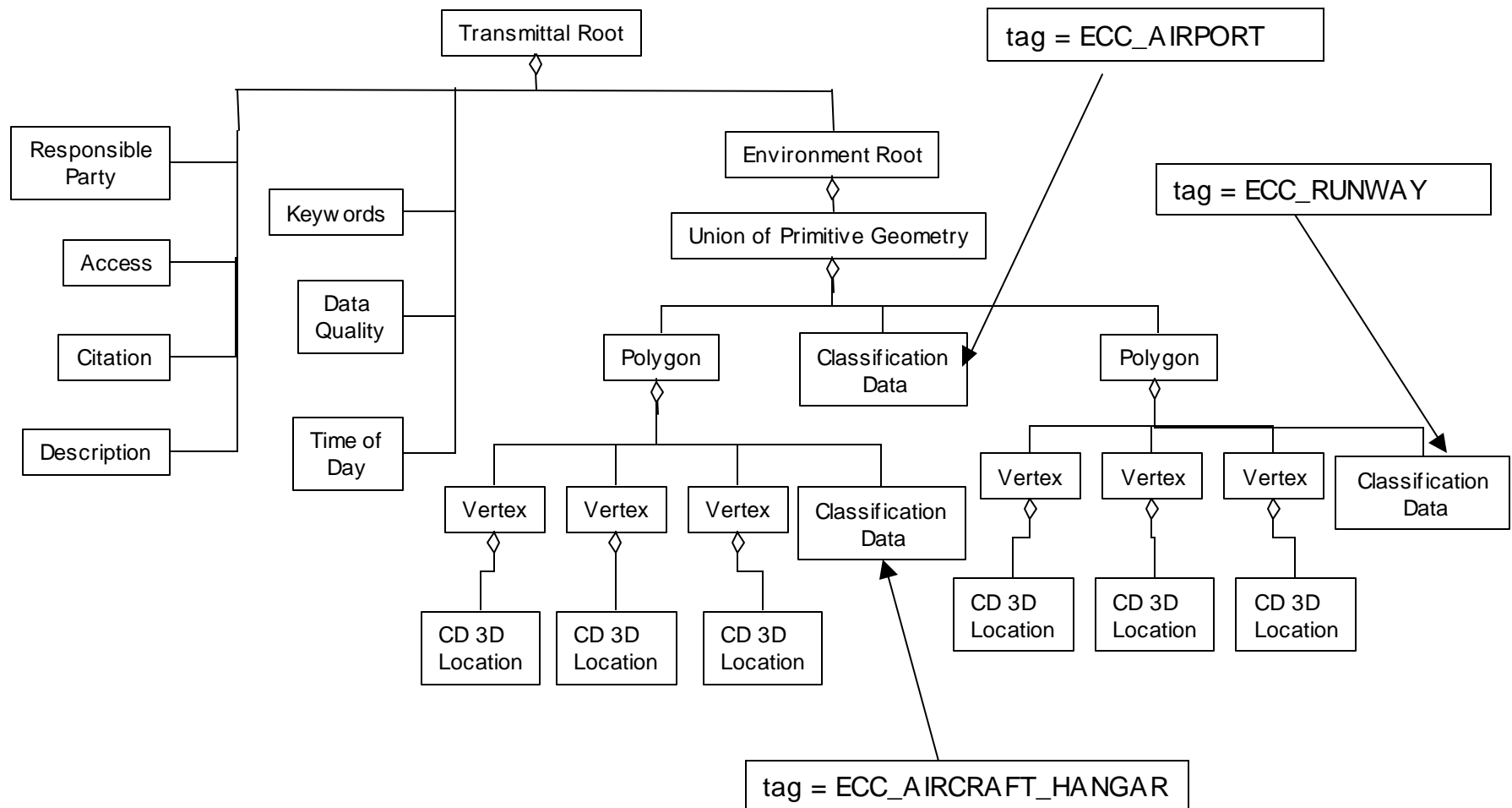
# What's in this transmittal?



- This transmittal contains an \_\_\_\_\_.



# What do we have here?





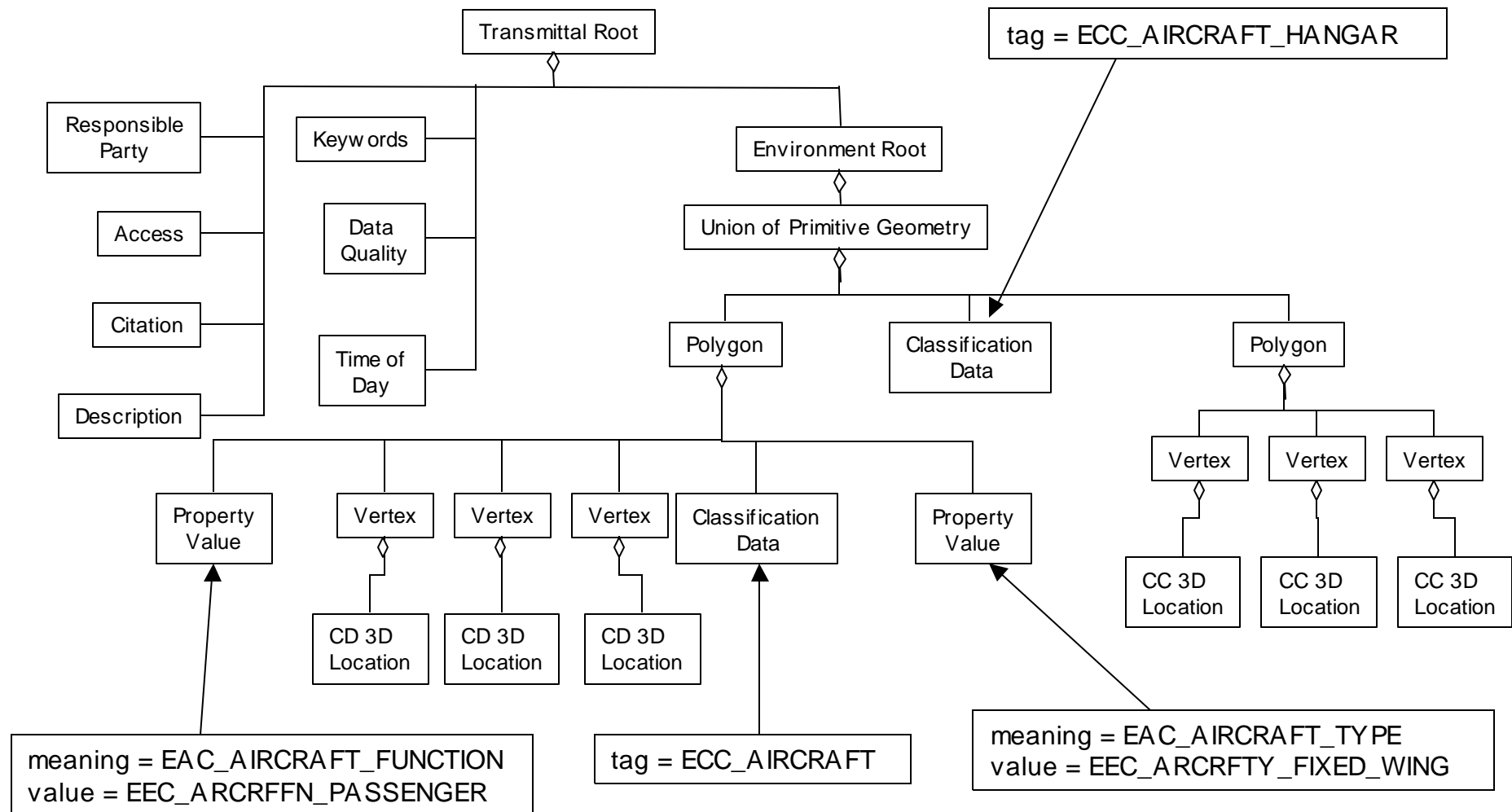
## Providing Attributes in the DRM

---

- **Once again, use the EDCS capabilities:**
  - EAC – Attributes
  - ESC – Scale
  - EUC - Units
- **An EDCS Attribute Code provides the semantic meaning of the property and its value in the DRM class <Property Value>**
- **A <Property Value> instance attached to some object specifies an attribute of that object tree**
- **Two pieces of information**
  - #1 is the meaning, i.e. the attribute
  - #2 is the value
- **Thus, using the same class, we can use any attribute!**



# What exactly do we have here?





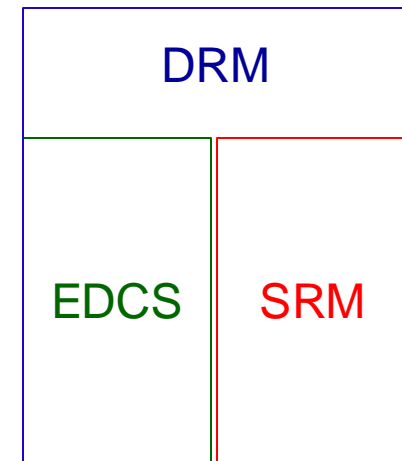
## Metadata in the DRM

- **Metadata in the DRM provides:**
  - Information about transmittal content designed for human users
  - Summary information that can also be used by consuming software
- **The first variety corresponds to the ISO metadata standard, examples:**
  - <Responsible Party>, <Access>, <Citation>, <Description>
- **The second variety consists of the "summary" classes:**
  - <Transmittal Summary>
  - <Environmental Domain Summary>
  - <SRF Summary>
  - <DRM Class Summary Item>
  - <Hierarchy Summary Item>
  - <Primitive Summary Item>



# DRM Summary

- Set of class definitions and constraints
- Set of defined relationships between classes, specified in UML
- Instanced as a transmittal
- <Feature> and <Geometry> primitives
- Uses SRM and EDCS
- Metadata at transmittal and object level
- Organizational capabilities provide flexibility





# DRM Tutorials

---

## **Fundamentals of the DRM**

**1:30 PM Tuesday, 6 January**

**Room: Diamond**

### **WHO SHOULD ATTEND:**

Environmental modelers interested in using SEDRI, software engineers who plan to implement applications based on SEDRI technologies, and those interested in gaining a better appreciation for the most fundamental SEDRI technology, the DRM

## **Advanced Application of the DRM**

**8:30 AM Wednesday, 7 January**

**Room: Diamond**

### **WHO SHOULD ATTEND:**

Environmental modelers and software engineers who are experienced users of SEDRI, interested in the newest developments as well as future advancements.



# **SEDRIS Transmittal Format (STF)**



## SEDRIIS Transmittal Format (STF)

---

- ***A file format developed to store SEDRIIS data***
- ***The data organization is derived directly from the DRM. Objects are stored and retrieved in direct 1-to-1 relationship with the data created by the producer***
- ***A binary format which is machine architecture and word order independent***
- **STF transmittals consist of multiple files but are named by one main 'root' file**
- **The SEDRIIS API Implementation provides the functionality of extraction and insertion DRM objects from the STF**



# Transmittal Format - Requirements

- ***Platform Independence:***
  - Both software and files
  - Adapts to platform's word order
- ***Fully Support the SEDRIS Data Representation Model:***
  - Full expressive power of the DRM
  - Data driven via DRM support functions
  - Completely loss-less with respect to objects instantiated by the data provider
- ***Space Efficient Media Storage:***

Minimal overhead with respect to the size of the SEDRIS objects stored
- ***Run-time Efficiency:***
  - Efficient with respect to both memory and processing time
  - Heavier emphasis on extraction performance than on insertion
- ***Insulate Developers from Implementation Details:***
  - Format and software can evolve independently from applications
- ***SW Implementation abstracts system services to simplify portability:***
  - Marshals system resources
  - Supports platform specific tuning to enhance performance



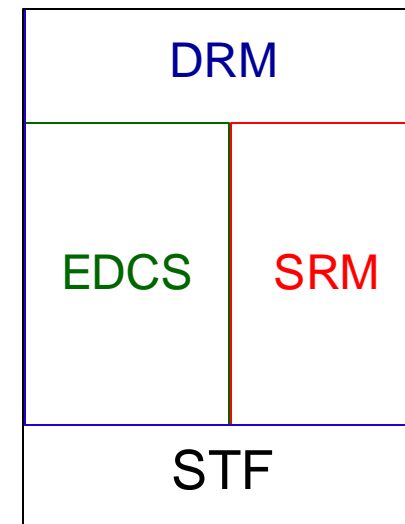
# STF Design Features

- ***File based storage on media:***

- Structure of data on physical media is the responsibility of the platform operating system, not the STF
- STF transmittals use can hierarchical directory structure to organize files

- ***In essence, implements a simple persistent object database system:***

- STF is intentionally *not* a full object- oriented database management system



- ***SW Implementation abstracts system services to simplify portability:***

- Marshals system resources
- Supports platform specific tuning to enhance performance

# **SEDRIS APIs**



## Definitions

---

- ***API (Application Program Interface):*** An encapsulation of functionalities common to many applications into reusable modules.
- ***API Implementation:*** The instantiation of an API's functionality in software that is bound to one or more software language.
- ***Transmittal:*** Environmental data realized as a collection of DRM-compliant instances conveyed using the SEDRIS API and/or STF.



# The SEDRI API

- The SEDRI API is an encapsulation of functionality which provides applications the ability to access DRM objects.
  - **Data Extraction** - provides searching methods to access DRM objects in a SEDRI transmittal.
  - **Data Insertion** - provides ability to create DRM objects in new or existing SEDRI transmittals.
  - **Data Representation Model** - provides access to meta-data about DRM classes, data types, and their allowable relationships.
- A set of function definitions (bindings).
- Why an API (not just a format specification) ?
  - It provides a consistent interface between a user's software application and SEDRI transmittals.
  - It decouples the user's application from the transmittal's format, allowing the DRM, the transmittal format, and the user's application to evolve relatively independently of each other.
  - It provides functions to simplify the navigation of complex transmittals.



## The APIs

---

- **The Transmittal Access API is used to insert and extract data into a Transmittal**
- **The DRM API is used to query information about the DRM such as:**
  - **Date elements, DRM classes, relationships, etc**
- **The SRM API provides coordinate conversion of data from one SRF to another**
- **The EDCS API provides access to the EDCS dictionaries of integer codes, labels and definitions**
- **The Transmittal Access API implementation relies on the DRM, SRM, and EDCS APIs**
- **The SRM and EDCS implementations are independent and can be used “stand alone”**





# The Transmittal Access API

---

- **The Transmittal Access API deals with transmittals and objects within those transmittal**
- **Objects that aggregate other objects are referred to as “aggregates”**
- **Objects that are aggregated by another object are referred to as “components”**
- **Objects that have an association relationship between them are referred to as “associates”**
- **Both a C and a C++ API are defined**
- **Every object has a unique string within a transmittal, referred to as the “object id”**
- **An object in one transmittal can reference objects in another transmittal through the use of “ITR”: Inter Transmittal Referencing**



# Transmittal Access API Functionality

---

- **Transmittal Functionality**
  - SE\_OpenTransmittalByFile
  - SE\_OpenTransmittalByName
  - SE\_CloseTransmittal
  - SE\_GetTransmittalFromObject
  - SE\_FreeTransmittal
  - SE\_GetTransmittalFile
  - SE\_GetTransmittalName
  - SE\_GetTransmittalVersionInformation
  - SE\_GetUniqueTransmittalID
  - SE\_SetTransmittalName
  - SE\_TransmittalsAreSame



# Transmittal Access API Functionality

---

- **Object instance functionality:**
  - SE\_GetRootObject
  - SE\_SetRootObject
  - SE\_CreateObject
  - SE\_CloneObject
  - SE\_RemoveFromTransmittal
  - SE\_FreeObject
  - SE\_GetIDForObject
  - SE\_GetObjectForID
  - SE\_GetPackedHierarchy
  - SE\_FreePackedHierarchy
  - SE\_GetRemainingObjectsList
  - SE\_FreeRemainingObjectsList
  - SE\_GetRemainingPackedHierarchiesList
  - SE\_FreeRemainingPackedHierarchiesList
  - SE\_GetDRMClass
  - SE\_GetFields
  - SE\_PutFields
  - SE\_ObjectsAreSame



# Transmittal Access API Functionality

---

- **Relationship Management:**
  - SE\_AddAssociateRelationship
  - SE\_AddComponentRelationship
  - SE\_RemoveAssociateRelationship
  - SE\_RemoveComponentRelationship
  - SE\_GetObjectReferenceCount
  - SE\_GetRelationCounts
- **Traversal/Extraction:**
  - SE\_GetAggregate
  - SE\_GetAssociate
  - SE\_GetComponent
  - SE\_InitializeAggregateIterator
  - SE\_InitializeAssociateIterator
  - SE\_InitializeComponentIterator
  - SE\_GetIterationLengthRemaining
  - SE\_GetNextObject
  - SE\_FreeIterator



# Transmittal Access API Functionality

---

- **Search Criteria**
  - SE\_CreateSearchFilter
  - SE\_FreeSearchFilter
  - SE\_CreateSpatialSearchBoundary
  - SE\_DetermineSpatialInclusion
  - SE\_FreeSpatialSearchBoundary
- **Data Tables**
  - SE\_GetDataTableData
  - SE\_PutDataTableData
- **Images**
  - SE\_GetImageData
  - SE\_PutImageData
- **Auto Conversion**
  - SE\_SetColourModel
  - SE\_SetSRFParameters



# Transmittal Access API Functionality

---

- **ITR functionality**
  - SE\_GetPublishedLabels
  - SE\_GetPublishedObjectList
  - SE\_GetReferencedTransmittalList
  - SE\_GetUnresolvedObjectFromPublishedLabel
  - SE\_ObjectIsPublished
  - SE\_ObjectIsResolved
  - SE\_PublishObject
  - SE\_ResolveObject
  - SE\_ResolveTransmittalName
  - SE\_UnpublishObject
- **Error handling functionality**
  - SE\_GetLastFunctionStats
  - SE\_SetSpecificCallback
  - SE\_SetFirstErrorMessage
  - SE\_SetGeneralCallback
  - SE\_SetSecondErrorMessage



# Basic Extraction Steps

## Logic

- Open a Transmittal
- Get the Root Object
- Create a Search Boundary
  - Create a Search Filter
  - Create an Iterator
    - Retrieve an object
    - Get the fields
    - Process the Object
    - Free the Object
  - Free the Iterator
  - Free the Search Filter
- Free the Search Boundary
- Free the Root Object
- Close the Transmittal

## Function Calls

- SE\_OpenTransmittalByFile
- SE\_GetRootObject
- SE\_CreateSpatialSearch Boundary
  - SE\_CreateSearchFilter
  - SE\_InitializeComponentIterator
    - SE\_GetNextObject
    - SE\_GetFields
  - SE\_FreeObject
  - SE\_Freeliterator
  - SE\_FreeSearchFilter
- SE\_FreeSpatialSearchBoundary
- SE\_FreeObject
- SE\_CloseTransmittal



# Basic Insertion Steps

## Logic

- Open a Transmittal
- Create the Root Object
- Set the Root Object Fields
- Set the Root Object for the Trans.
- Create an Object
  - Set the Object Fields
  - Add as Component of Root
    - Create an Object
    - Set the Object Fields
    - Add as Component
    - Free the Object
  - Free the Object
- Free the Root Object
- Close the Transmittal

## Function Calls

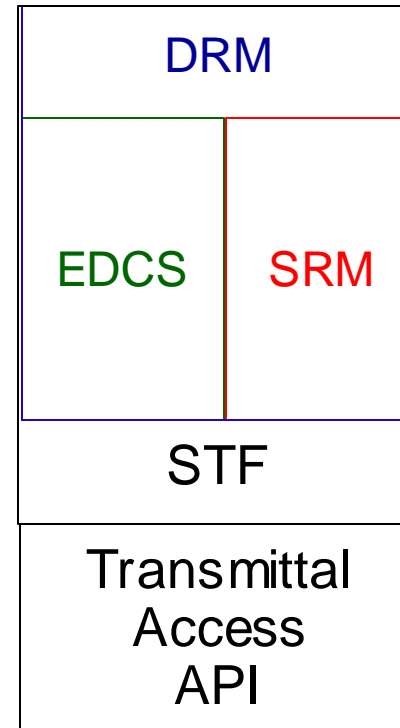
- SE\_OpenTransmittalByFile
- SE\_CreateObject
- SE\_SetFields
- SE\_SetRootObject
- SE\_CreateObject
  - SE\_SetFields
  - SE\_AddComponentRelationship
    - SE\_CreateObject
    - SE\_SetFields
    - SE\_AddComponentRelationship
    - SE\_FreeObject
  - SE\_FreeObject
- SE\_FreeObject
- SE\_CloseTransmittal





# Transmittal Access API Summary

- A handful of concepts
  - Transmittal
  - Objects
  - Object Ids
  - Traversal
  - Iterators
- Provides all functionality necessary to retrieve the data in a transmittal
- Provides all functionality necessary to create a transmittal as well as modify it
- Provides helper functions such as enumerations to string





# Transmittal Access API Tutorial

---

## Fundamentals for Accessing Transmittals

**1:30 PM Tuesday, 6 January**

**Room: Emerald**

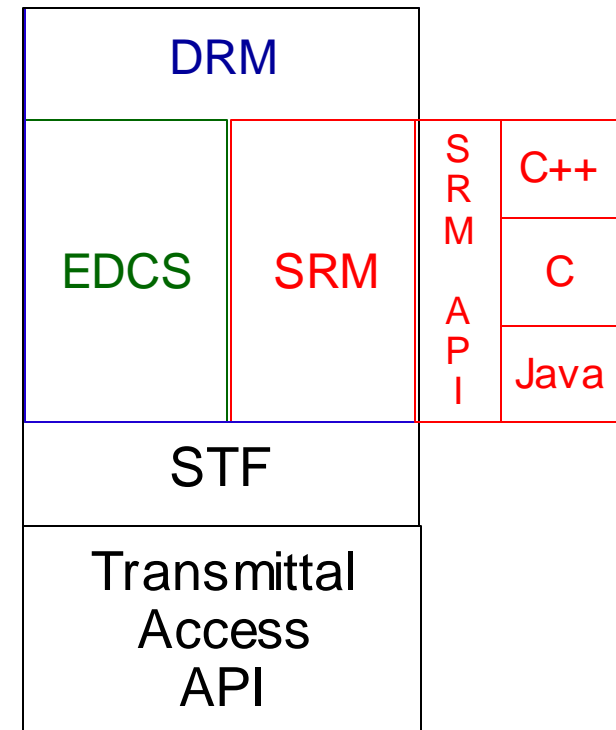
### WHO SHOULD ATTEND

Software engineers who intend to design and implement SEDRIIS-based applications, or those interested in learning the basic functionality of the SEDRIIS APIs.



# The SRM API

- The SRM API deals with SRFs, coordinates, directions, and orientation constructs
- Data structures are defined for the instantiation of each concept
- SRF constructs manage coordinates, directions, and orientations constructs
- Functionality provided for:
  - SRF, coordinate, direction, & orientation construct creation
  - Inter SRF conversion for coordinates, direction, & orientation constructs
  - Specific calculations
    - ✍ ConvergenceOfTheMeridian
    - ✍ GeodesicDistance
    - ✍ EuclideanDistance
- ✍ Multiple language implementations
  - ✍ C, C++, & Java





# SRM Tutorials

---

## **SRM for Programmers**

### **4 PM Tuesday, 6 January**

### **Room: Sapphire**

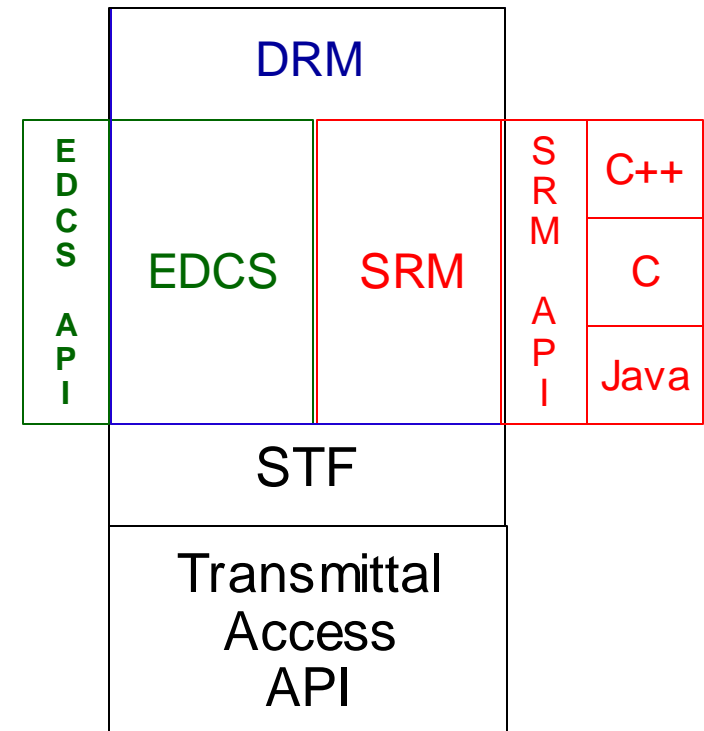
#### **WHO SHOULD ATTEND:**

Those interested in using the SRM API to perform operations on spatial data, including coordinate conversions and datum shifts.



# The EDCS API

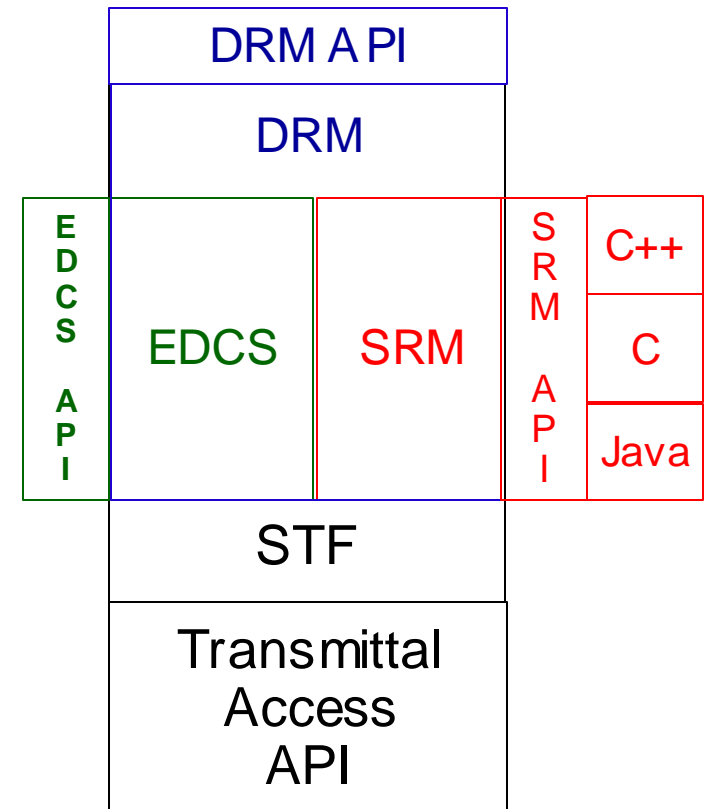
- Functions for converting character strings to enumerated types
- Functions for converting simple and structured types to character strings
- Functions for validating simple and structured types
- Functions for comparing data structures (qsort compatible)
- Function for converting quantity values between units
- 27 Functions for interacting with the 9 EDCS Dictionaries (3 each)
  - For retrieving the EDCS dictionary entry by code
  - For converting EDCS labels to codes
  - For converting C binding “Symbolic Constants” (mnemonics)





# The DRM API

- **Functionality to query:**
  - DRM types
  - DRM classes
  - DRM inheritance
  - Allowable relationships between DRM Classes
  - Fields and field elements
- **Functionality for converting data types to character strings**
- **Functionality for validating simple and structured types**
- **Allows for software to query for DRM definitions**
- **Also provides functionality for converting between color models**



# **SEDRIS SDKs**



# Component SDKs

- **EDCS SDK**
  - Standalone EDCS implementation
  - Build environment to compile and link included software
  - Utilities to test executables following builds
  - HTML Documentation:
    - Reference Manual
    - Migration Guide for upgrading from previous releases
    - Mapping utilities from other standards e.g. FACC 2.1
- **SRM SDKs**
  - Standalone SRM implementation
  - Build environment to compile and link included software
  - Utilities to test executables following builds
  - HTML Documentation:
    - Reference Manuals for: EDCS, SRM, DRM, All APIs
    - Migration Guide for upgrading from previous releases
  - Individual SDK for Java, C, and C++





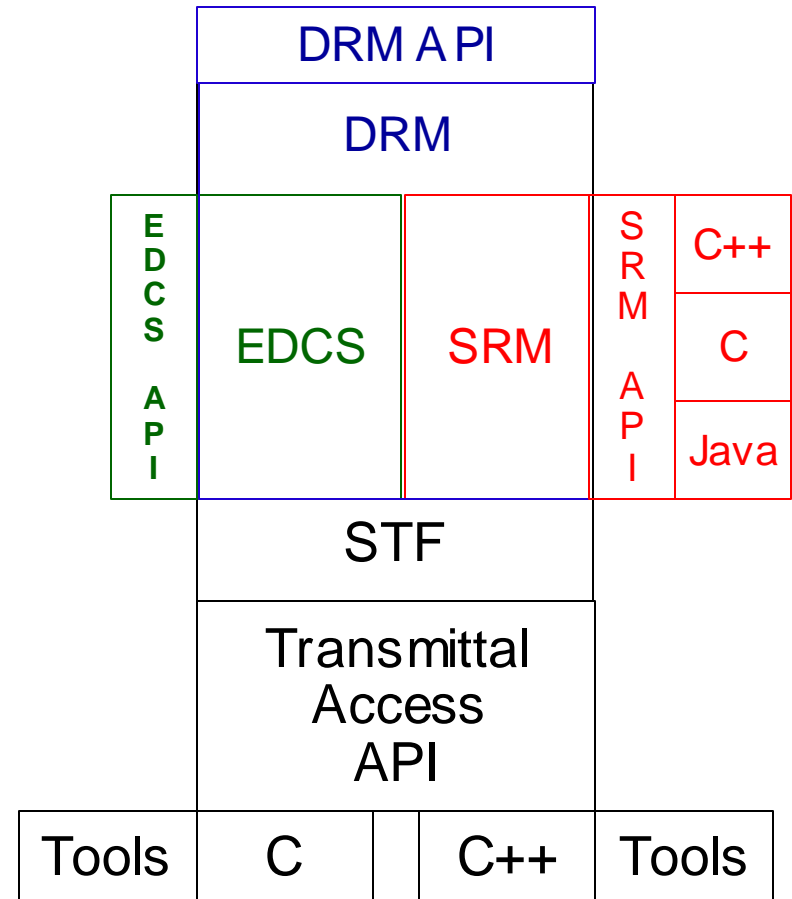
# SEDRIIS SDK

- **Source Code to Reference Implementation of:**
  - EDCS API
  - SRM API
  - DRM API
  - Transmittal Access API implemented over STF
- **Standard Tools**
  - Depth – Traverses an STF transmittal and produce an ASCII dump
  - Model Viewer – OpenGL based viewer for Model Geometry
  - Syntax Checker - Traverses a transmittal and validates object relationships against DRM
  - Rules Checker – Traverses transmittal and validates content against DRM constraints
- **Build environment to compile and link included software**
- **Utilities to test executables following builds**
- **Sample transmittals**
- **HTML Documentation:**
  - Build Kit: build & install
  - Reference Manuals for: EDCS, SRM, DRM, All APIs
  - Migration Guide for upgrading from previous releases
  - Technical Guides for Special Topics
  - User Guides for Standard Tools



# SEDRIIS SDK

- Current version is SEDRIIS SDK Release 3.1.2
- Next release is 4.0, adding
  - C++ SRM
  - Java SRM
  - C++ Transmittal Access API
- Release numbering includes major, minor, and maintenance version
- Releases with major and minor versions are compatible
- Release 4.0 is currently in testing and scheduled to be released in February
- 4.0 functionality is presented at this STC





# SEDRIIS SDK Tutorial

---

## Advanced Use of the SEDRIIS SDK

**8:30 AM Wednesday, 7 January**

**Room: Emerald**

### WHO SHOULD ATTEND

**Users of the SEDRIIS SDK who want hands on experience with developing software using the SDK. This tutorial will focus on developing software using the Transmittal Access C++ API.**

# Conclusion



# Solving Environmental Data Challenges

---

- A mechanism to access and interact with any data sets or data collectors through a robust software interface
- Capture and communicate the resulting data in a persistent, efficient, and platform independent format designed to handle large and distributed data sets
- Tools to manipulate, evaluate, visualize, or analyze the data
- Automatically evaluate and validate data sets against stated requirements
- **The API implementations do this**
- **The STF is designed for this**
- **An array of powerful tools and utilities**
- **Transmittal Content Requirements Specification (TCRS)**



# Solving Environmental Data Challenges

---

- Representation of location for the various coordinate systems (spatial reference frames), local or global, that will be “natural” for individual systems or sub-systems
- Accurate, efficient, and fast conversion of location data between different spatial reference frames
- Comprehensive dictionary of terms that not only deals with terrain data, but also atmosphere, ocean, littoral, and space data. And is also extensible in a predictable and supported manner
- A representation schema that can handle any resolution, type, organization, and extent of environmental data through a uniform approach for all domains of the environment
- **The SRM is designed for this**
- **The SRM implementation does this**
- **The EDCS is designed for this**
- **The DRM is designed for this**

**Maximize Your STC Experience!**

**Any Questions?**



# Tutorial Answers

---

- Slide 6:
  - 5 exabytes => 5,000,000 terrabytes
  - 500,000 U.S. Libraries of Congress
  - 800 MB
  - Doubling every 3 years
- Slide 66:
  - Yes
  - No; the <3D Location> and <Volume Extent> instances must be concrete class instances
- Slide 67:
  - Not Enough Info; is <Union of Primitive Geometry> an <Aggregate Geometry>? Yes, so the instance diagram is valid.
  - No, remove <Classification Data>





# Tutorial Answers

---

- **Slide 68:**
  - The first <Sound Volume> should have a concrete <Volume Extent>
  - The <CC 3D Location> under the <Parallelepiped Volume Extent> should be aggregated by the <Sound Volume> parent
  - The <Cylindrical Volume Extent> has an extra <Reference Vector>
  - The <Parallelepiped Volume Extent> is missing one <Reference Vector>
- **Slide 69:**
  - No, need an additional <CD 3D Location>
  - No, the <CD Surface Location> cannot be shared; the <Cylindrical Volume Extent> needs a <3D Location> concrete subclass instance
- **Slide 88:**
  - EDCS
- **Slide 90:**
  - The EDCS definition of an airport.