Apache SIS for earth observation

Martin Desruisseaux

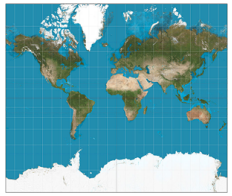
GEOMATYS

Agenda

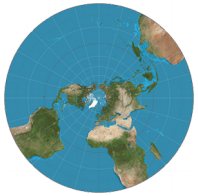
- Open Geospatial Consortium
- Data discovery
  - NetCDF
  - Landsat 8
  - Other formats
- Units of measurement
- Coordinate Reference System
- Use for planetary bodies
- Next developments

Geospatial can be complex

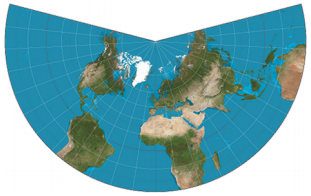
- Various map projections, geodetic datum, axis conventions…



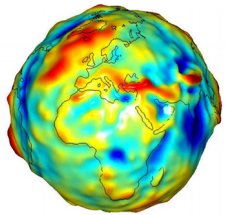
Mercator



Stereographic



Lambert conic conformal

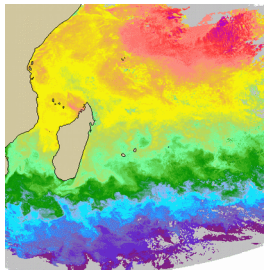


Geoid

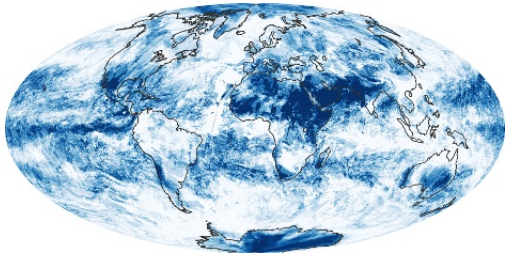
Credit: NASA/JPL/University of Texas

- Various phenomenons, not always two-dimensional

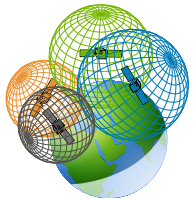
Credit: NASA Earth Observatory



Credit: IRD

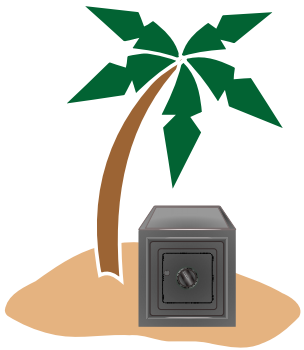


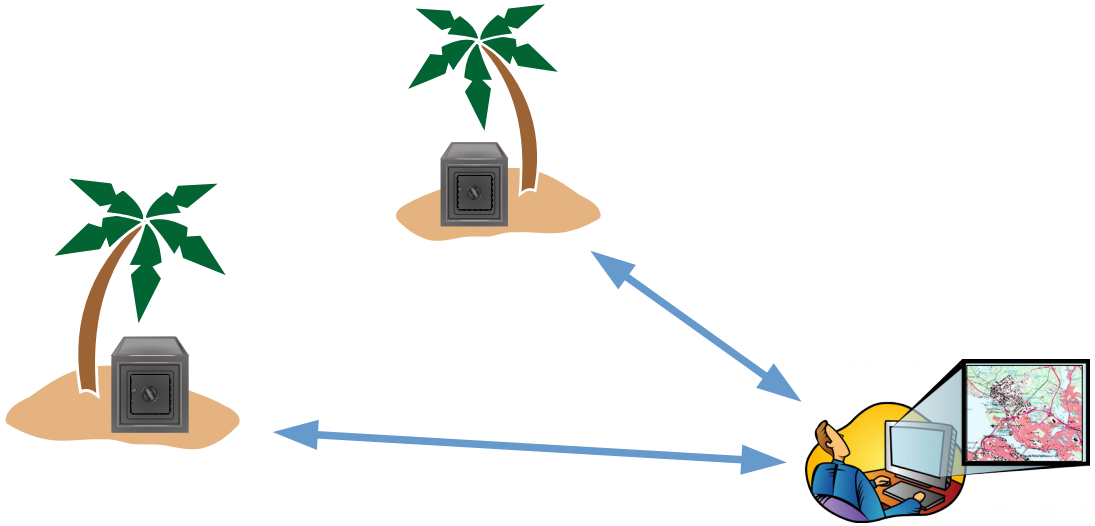
Credit: NASA Earth Observatory



Barrier in geospatial data exchange

- Incompatible data
- Incompatible system
- Data fragmentation
- Redundancy





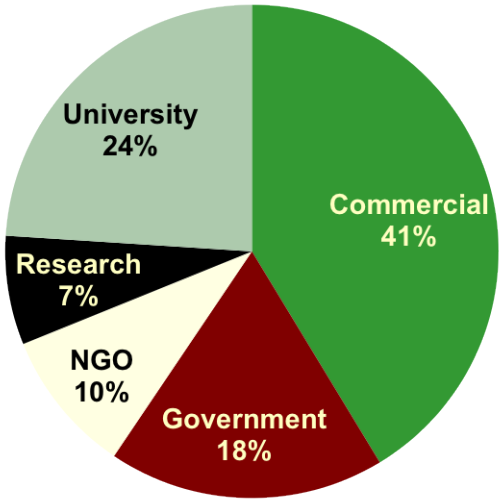
Need for:

- Share catalogs and maps on the web
- Deliver data to different systems easily
- Common language for geospatial data and services

Open Geospatial Consortium (OGC)

Not-for-profit, international voluntary consensus standards organization; leading development of geospatial standards.

- Founded in 1994
- 520+ members and growing
- 50 standards
- Hundreds of product implementations
- Broad user community implementation worldwide
- Collaborative activities with professional associations



http://www.opengeospatial.org/

What open standards give

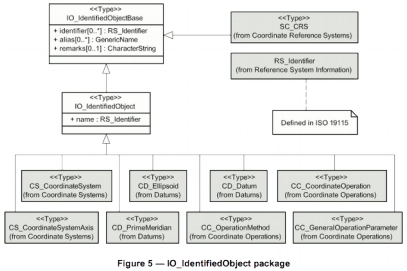
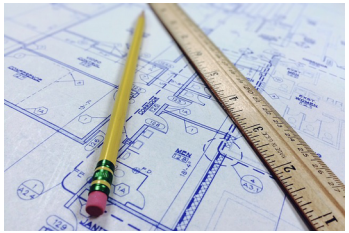


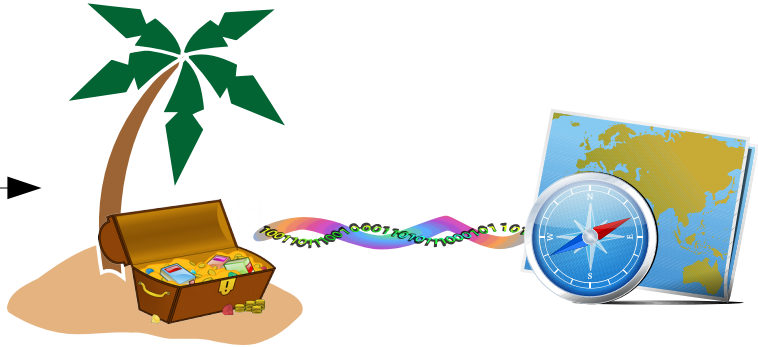
Figure 5 — IO_IdentifiedObject package

- UML as blueprint for applications



```
<complexType name="GeodeticCRSType">
 <complexContent>
  <extension base="AbstractCRSType">
   <sequence>
    <choice>
     <element ref="ellipsoidalCSProperty"/>
     <element ref="cartesianCSProperty"/>
     <element ref="sphericalCSProperty"/>
    </choice>
    <element ref="geodeticDatumProperty"/>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```

- Data exchange formats
- Web services API



Standards for Java developers

Do not use API generated from XSD!

```
CoordinateSystem cs;
if (crs instanceof GeodeticCRS) {
    GeodeticCRS geodeticCRS = (GeodeticCRS) crs;
    cs = geodeticCRS.getEllipsoidalCS();
    if (cs == null) {
        cs = geodeticCRS.getSphericalCS();
        if (cs == null) {
            cs = geodeticCRS.getCartesianCS();
        }
    }
} else if (crs instanceof VerticalCRS) {
    VerticalCRS verticalCRS = (VerticalCRS) crs;
    cs = verticalCRS.getVerticalCS();
} else if (crs instanceof EngineeringCRS) {
    EngineeringCRS engineeringCRS = (EngineeringCRS) crs;
    cs = engineeringCRS.getEllipsoidalCS();
    if (cs == null) {
        cs = engineeringCRS.getSphericalCS();
        if (cs == null) {
            cs = engineeringCRS.getCartesianCS();
            if (cs == null) {
                cs = engineeringCRS.getPolarCS();
                if (cs == null) {
                    cs = engineeringCRS.getCylindricalCS();
                }
            }
        }
    }
} else // etc.
```

Use API derived by human

```
CoordinateSystem cs = crs.getCoordinateSystem();
```

http://www.geoapi.org

Why?

	Java	XSD	JAXB
Union	X	ok	X
Type covariance	ok	X	X

+ “property versus type” duality in GML

Apache Spatial Information System (SIS)

Latest release: 0.7 (may 2016)
Current development: 0.8

ISO	OGC	Topic	GeoAPI	Apache SIS
19103		Conceptual schema language	3.0.0	0.3
19115		Metadata (including imagery and gridded data extension)	3.0.0	0.3 (updated in 0.5)
19139		Metadata — XML schema		0.3
19111	08-015	Spatial referencing by coordinates	3.0.0	0.4, 0.5, 0.6, 0.7
19162	12-063	Well Known Text (WKT) representation of reference systems		0.6 (updated in 0.7)
19136	07-036	Geographic Markup Language (GML)		0.6 (updated in 0.7)
19109		Rules for application schema (Features)	3.1-SNAPSHOT	0.5
19107		Feature geometry (1 to 3 dimensional)	pending	
19123	07-011	Coverage geometry and functions	pending	Pending port from the Geotk project.
19156	10-004	Observation and measurement	pending	
13249		SQL spatial		
	12-168	Catalog Services (CSW)		Google Summer of Code
19128	06-042	Web Map Service (WMS)		Pending port from the Constellation project.
19142	09-025	Web Feature Service (WFS)		

Data discovery

- ISO 19115
- Examples
  - NetCDF
  - Landsat 8
  - Other formats

ISO 19115 — geographic metadata (1/2)

Metadata

Data identification

Citation
- Titles
- Authors (creator, contributor…)

Data format

Spatiotemporal extent
- Geographic bounding box
- Vertical and temporal ranges

Resolution

Content information
- Illumination elevation & azimuth angles
- Cloud cover percentage

Attribute (band) group
- Content type (physical measurement, …)

Attribute (band)
- Description (coastal aerosol, …)
- Peak response in nanometres
- Transfer function

More attributes (bands)…

ISO 19115 — geographic metadata (2/2)

Metadata

Spatial representation

(more on it later)

Acquisition

- Platform & instruments
- Operation (status, events, …)

Distribution

- Format
- Digital transfer options

Lineage

- Processors (organization, …)
- Process steps (inputs, algorithm, …)

Data quality

- Completeness
- Consistency (logical, thematic, …)

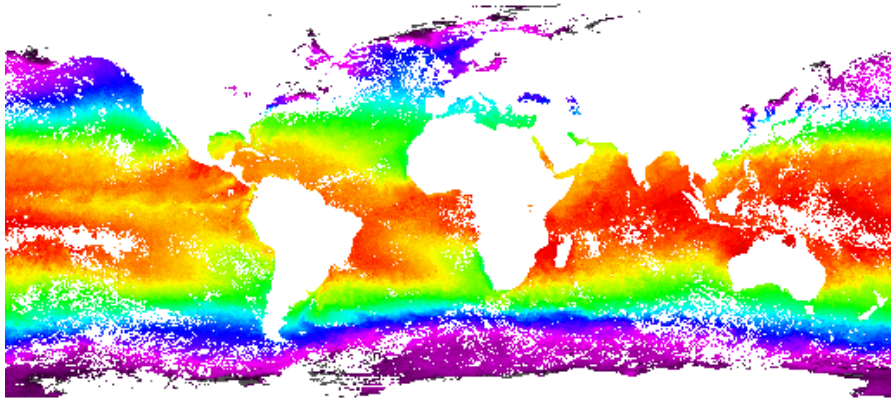
Maintenance

- Scope (dataset, software, …)
- Dates & update frequency

Constraints (legal, security, …)

NetCDF

(since SIS 0.3)



- Binary encoding of arrays of scientific data
- Used for space and time-varying phenomena
- Attributes defined by various conventions
  - Climate and Forecast (CF-convention)
  - Attribute Convention for Dataset Discovery (ACDD)
- Mapping to ISO 19115 documented on NOAA wiki

```
netcdf SST {
dimensions:
    lat = 1800 ;
    lon = 3600 ;
    time = 21 ;
variables:
    float lat(lat) ;
        lat:long_name = "latitude" ;
        lat:units = "degrees_north" ;
    float lon(lon) ;
        lon:long_name = "longitude" ;
        lon:units = "degrees_east" ;
    double time(time) ;
        time:long_name = "valid time" ;
        time:units = "hours since 1992-1-1" ;
    short SST(time, lat, lon) ;
        SST:long_name = "Sea water temperature" ;
        SST:units = "degC" ;
        SST:scale_factor = 0.001 ;
        SST:add_offset = -5.0 ;
        SST:_FillValue = 32767s ;
    …
// global attributes:
        :title = "Sea Surface Temperature" ;
        :summary = "SST Global 0.1 x 0.1 degree data" ;
        :keywords = "EARTH SCIENCE > Oceans > Ocean
                Temperature > Sea Surface Temperature" ;
        :date_created = "2005-09-22T00:00" ;
        :geospatial_lat_min = "-90.0" ;
        :geospatial_lat_max = "90.0" ;
        :geospatial_lon_min = "-180.0" ;
        :geospatial_lon_max = "180.0" ;
    …
```

NetCDF to ISO 19115

(since SIS 0.3)

```
Metadata
  ├─Spatial representation info
  │   ├─Number of dimensions……… 3
  │   ├─Axis dimension properties (1 of 3)
  │   │   ├─Dimension name……… Column
  │   │   └─Dimension size……… 3600
  │   ├─Axis dimension properties (2 of 3)
  │   │   ├─Dimension name……… Row
  │   │   └─Dimension size……… 1800
  │   ├─Axis dimension properties (3 of 3)
  │   │   ├─Dimension name……… Time
  │   │   └─Dimension size……… 21
  │   ├─Cell geometry……… Area
  │   └─Transformation parameter availability…… false
  ├─Identification info
  │   ├─Citation
  │   │   ├─Title……… Sea Surface Temperature
  │   │   ├─Date
  │   │   │   ├─Date……… Sep 22, 2005 2:00:00 AM
  │   │   │   └─Date type……… Creation
  │   │   └─Cited responsible party
  │   │       ├─Party
  │   │       │   └─Name……… NOAA
  │   │       └─Role……… Originator
  │   ├─Abstract……… SST Global 0.1 x 0.1 degree data
  │   ├─Descriptive keywords
  │   │   ├─Keyword……… EARTH SCIENCE > Oceans > …
  │   │   └─Type……… Theme
  │   ├─Spatial representation type……… Grid
  │   └─Extent
  │       └─Geographic element
  │           ├─West bound longitude……… 180°W
  │           ├─East bound longitude……… 180°E
  │           ├─South bound latitude……… 90°S
  │           ├─North bound latitude……… 90°N
  │           └─Extent type code……… true
  ├─Content info
  │   └─Attribute group
  │       └─Attribute
  │           ├─Sequence identifier……… SST
  │           ├─Units……… °C
  │           ├─Scale factor……… 0.001
  │           ├─Offset……… -5
  │           ├─Transfer function type……… Linear
  │           └─Description……… Sea water temperature
  ├─Metadata scope
  │   └─Resource scope……… Dataset
  └─(…snip…)
```

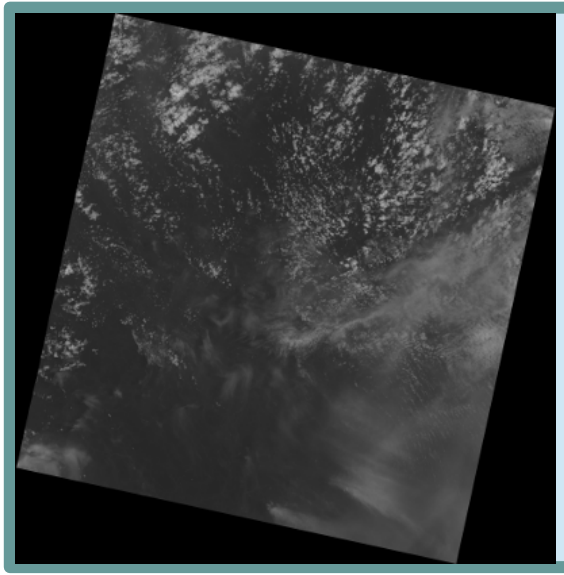
```
netcdf SST {
dimensions:
   lat = 1800 ;
   lon = 3600 ;
   time = 21 ;
variables:
   float lat(lat) ;
      lat:long_name = "latitude" ;
      lat:units = "degrees_north" ;
   float lon(lon) ;
      lon:long_name = "longitude" ;
      lon:units = "degrees_east" ;
   double time(time) ;
      time:long_name = "valid time" ;
      time:units = "hours since 1992-1-1" ;
   short SST(time, lat, lon) ;
      SST:long_name = "Sea water temperature" ;
      SST:units = "degC" ;
      SST:scale_factor = 0.001 ;
      SST:add_offset = -5.0 ;
      SST:_FillValue = 32767s ;
   …
// global attributes:
      :title = "Sea Surface Temperature" ;
      :summary = "SST Global 0.1 x 0.1 degree data" ;
      :keywords = "EARTH SCIENCE > Oceans > Ocean
            Temperature > Sea Surface Temperature" ;
      :date_created = "2005-09-22T00:00" ;
      :geospatial_lat_min = "-90.0" ;
      :geospatial_lat_max = "90.0" ;
      :geospatial_lon_min = "-180.0" ;
      :geospatial_lon_max = "180.0" ;
   …
```

Landsat 8

(since SIS 0.8)

11 bands:

- Coastal aerosol
- Blue
- Green
- Red
- Near-infrared
- Short wavelength infrared 1
- Short wavelength infrared 2
- Panchromatic
- Cirrus
- Thermal infrared sensor 1
- Thermal infrared sensor 2



```
GROUP = L1_METADATA_FILE
  GROUP = METADATA_FILE_INFO
    ORIGIN = "Image courtesy of the U.S. Geological Survey"
    REQUEST_ID = "0501403126384_00011"
    LANDSAT_SCENE_ID = "LC81230522014071LGN00"
    FILE_DATE = 2014-03-12T06:06:35Z
    STATION_ID = "LGN"
    PROCESSING_SOFTWARE_VERSION = "LPGS_2.3.0"
  END_GROUP = METADATA_FILE_INFO
  GROUP = PRODUCT_METADATA
    DATA_TYPE = "L1T"
    ELEVATION_SOURCE = "GLS2000"
    OUTPUT_FORMAT = "GEOTIFF"
    SPACECRAFT_ID = "LANDSAT_8"
    SENSOR_ID = "OLI_TIRS"
    WRS_PATH = 123
    WRS_ROW = 52
    NADIR_OFFNADIR = "NADIR"
    TARGET_WRS_PATH = 123
    TARGET_WRS_ROW = 52
    DATE_ACQUIRED = 2014-03-12
    SCENE_CENTER_TIME = 03:02:01.5339408Z
    …
```

Image by NASA / USGS

Landsat 8 to ISO 19115

(since SIS 0.8)

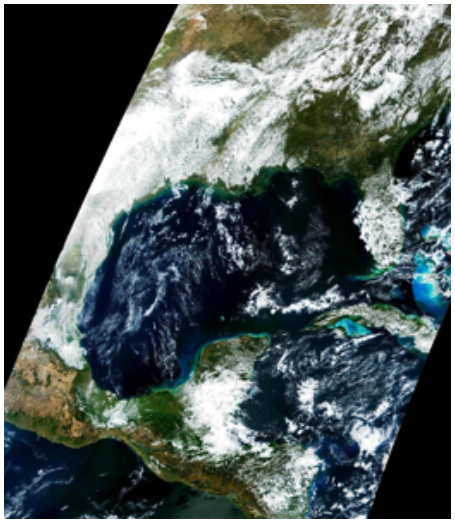
```
Metadata
  ├─Language……  English
  ├─Spatial representation info (1 of 3)
  │   ├─Axis dimension properties (1 of 2)
  │   │   ├─Dimension name……  Sample
  │   │   └─Dimension size……  15,241
  │   └─Axis dimension properties (2 of 2)
  │       ├─Dimension name……  Line
  │       └─Dimension size……  15,581
  ├─(…snip…)
  ├─Reference system info……  EPSG:WGS 84 / UTM zone 49N
  ├─Identification info
  │   ├─Citation
  │   │   ├─Date……  Mar 12, 2014 6:06:35 AM
  │   │   ├─Identifier……  LC81230522014071LGN00
  │   │   └─Cited responsible party……  U.S. Geological Survey
  │   ├─Resource format……  GeoTIFF Coverage Encoding Profile
  │   ├─Spatial resolution……  15
  │   └─Extent
  │       └─Geographic element
  │           ├─West bound longitude……  108°20′10.464″E
  │           ├─East bound longitude……  110°26′39.66″E
  │           ├─South bound latitude……  10°29′59.604″N
  │           └─North bound latitude……  12°37′25.716″N
  ├─Content info
  │   ├─Illumination elevation angle……  58.809
  │   ├─Illumination azimuth angle……  116.887
  │   ├─Cloud cover percentage……  8.34
  │   ├─Attribute group (1 of 3)
  │   │   ├─Content type……  Physical measurement
  │   │   ├─Attribute (1 of 8)
  │   │   │   ├─Peak response……  433
  │   │   │   ├─Scale factor……  0.013
  │   │   │   ├─Offset……  -63.594
  │   │   │   ├─Bound units……  nm
  │   │   │   ├─Description……  Coastal Aerosol
  │   │   │   └─Name……  LC81230522014071LGN00_B1.TIF
  │   │   └─(…snip…)
  │   └─(…snip…)
  ├─Acquisition information
  │   ├─Operation
  │   │   ├─Status……  Completed
  │   │   ├─Type……  Real
  │   │   └─Significant event……  Acquisition
  │   │       └─Time……  Mar 12, 2014 3:02:01 AM
  │   └─Platform
  │       ├─Identifier……  LANDSAT_8
  │       └─Instrument……  OLI_TIRS
  └─(…snip…)
```

```
GROUP = L1_METADATA_FILE
  GROUP = METADATA_FILE_INFO
    ORIGIN = "Image courtesy of the U.S. Geological Survey"
    REQUEST_ID = "0501403126384_00011"
    LANDSAT_SCENE_ID = "LC81230522014071LGN00"
    FILE_DATE = 2014-03-12T06:06:35Z
    STATION_ID = "LGN"
    PROCESSING_SOFTWARE_VERSION = "LPGS_2.3.0"
  END_GROUP = METADATA_FILE_INFO
  GROUP = PRODUCT_METADATA
    DATA_TYPE = "L1T"
    ELEVATION_SOURCE = "GLS2000"
    OUTPUT_FORMAT = "GEOTIFF"
    SPACECRAFT_ID = "LANDSAT_8"
    SENSOR_ID = "OLI_TIRS"
    WRS_PATH = 123
    WRS_ROW = 52
    NADIR_OFFNADIR = "NADIR"
    TARGET_WRS_PATH = 123
    TARGET_WRS_ROW = 52
    DATE_ACQUIRED = 2014-03-12
    SCENE_CENTER_TIME = 03:02:01.5339408Z
    …
```

Implementation in Apache SIS 0.8 by Thi Phuong Hao Nguyen and Minh Chinh Vu (Google Summer of Code)

Other formats

MODIS
(SIS implementation by Google Summer of Code, not yet merged)



GeoTIFF (in progress for SIS 0.8)
- Discovery metadata
- Coordinate Reference System

GPS Exchange Format (GPX)
(in progress for SIS 0.8)

Apache SIS API

Read (file format detected automatically)

```
Metadata metadata;
try (DataStore ds = DataStores.open(new File("myFile.txt"))) {
    metadata = ds.getMetadata();
}
```

Use or modify (~ 400 properties)

```
for (Identification id : metadata.getIdentificationInfo()) {
    for (Extent ex : id.getExtents()) {
        // …etc…
    }
}
```

There is various resources on the web providing guidance on ISO 19115 metadata. See for example NOAA wiki.

Print as a tree (Unicode & monospaced font)

```
System.out.println(metadata);
```

Export to XML (ISO 19139)

```
XML.marshal(metadata, myOutputStream);
```

Application examples

(command-line demo)

Units of measurement

- JSR-363
- Apache SIS implementation

JSR-363 — Units of Measurement API

- Final approval in September 2016
- 1 `Unit` interface
- 35 `Quantity` interfaces (`Angle`, `Length`, `Time`, `Speed`, *etc.*)
- Arithmetic operations on units
- Conversions of floating point values

```
Unit<Length> METRE  = ...;
Unit<Time>   SECOND = ...;
Unit<Speed>  METRE_PER_SECOND = METRE.divide(SECOND).asType(Speed.class);
```

```
Unit<Length>     myUnit = ...;
double distanceInMyUnit = ...;
UnitConverter converter = myUnit.getConverterTo(METRE);
double distanceInMetres = converter.convert(distanceInMyUnit);
```

Apache SIS implementation of JSR-363

- Convenience static constants for units of geospatial interest
  Example: `METRE`, `KILOMETRE`, `NAUTIC_MILE`, `US_SURVEY_FOOT`, *etc.*
- Decimal and sexagesimal degrees
- EPSG codes and URN in OGC namespace
  Example: "`urn:ogc:def:uom:EPSG::9001`" identifies "metre".
- Lenient parsing of unit syntax that differ from ISO standards
  Example: units in NetCDF files.
- Recognize existing units after arithmetic operations

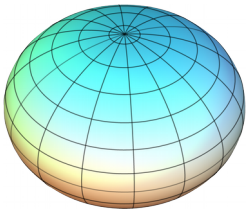
```
Unit<?> myUnit = Units.valueOf("1000.N.m/s");
System.out.println(myUnit);                                  // prints "kW"
```

Coordinate Reference Systems

- Get a Coordinate Reference System
- Get a Coordinate Operation
- Verify domain of validity
- Transform points and envelopes

Coordinate Reference Systems

In EPSG dataset 8.9.5:

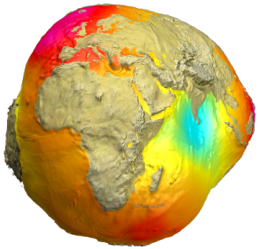


48 ellipsoids

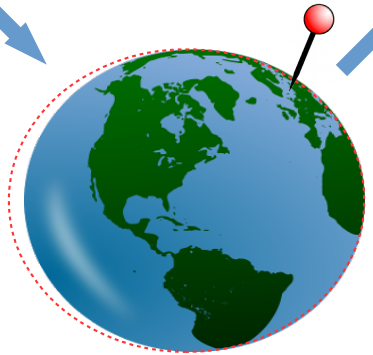
466 two-dimensional geographic coordinate reference systems

Latitudes and longitudes without CRS are ambiguous

3 km error in some parts of the world



1 planet



473 geodetic datums

4434 projected coordinate reference systems

Use definitions from a registry

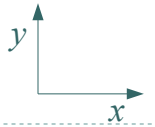
1) Search online: http://epsg-registry.org/
2) Give the EPSG code to Apache SIS
3) Verify if the CRS is the expected one

http://spatialreference.org/ and http://epsg.io/ are not official sources of EPSG definitions. They differ in axis order and other aspects.

```
CoordinateReferenceSystem myDataCRS = CRS.forCode("EPSG:3395");
```

Apache SIS 0.7 support about 5600 codes
http://sis.apache.org/tables/CoordinateReferenceSystems.html

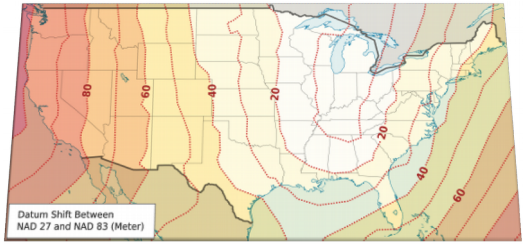
Well Known Text (WKT) format *version 2* (ISO 19162)



```
COMPOUNDCRS["NAD83 UTM 10 + NAVD88",
  PROJCRS["NAD83 UTM 10",
      BASEGEODCRS["NAD83",
        DATUM["North American Datum 1983",
          ELLIPSOID["GRS 1980", 6378137, 298.257222101]]]
      CONVERSION["UTM zone 10N",
        METHOD["Transverse Mercator", ID["EPSG", 9807]],
        PARAMETER["Longitude of natural origin", -123],
        PARAMETER["Scale factor", 0.9996],
        PARAMETER["False easting", 500000]],
      CS[Cartesian, 2],
        AXIS["Easting (E)", east],
        AXIS["Northing (N)", north],
        LENGTHUNIT["metre", 1]],
    VERTCRS["NAVD88",
      VDATUM["North American Vertical Datum 1988"],
      CS[vertical, 1],
        AXIS["gravity-related height (H)", up],
        LENGTHUNIT["metre", 1]]]
```

```
CoordinateReferenceSystem myDataCRS = CRS.fromWKT("COMPOUNDCRS[…]");
```

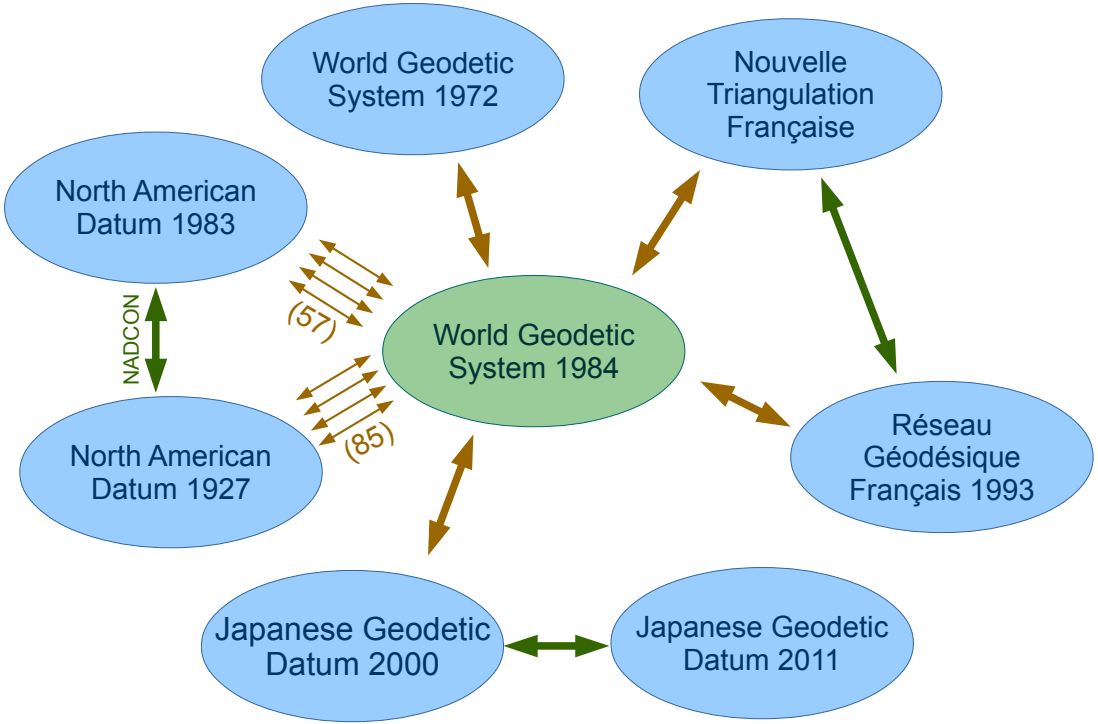
Relationship between CRS



- NAD83 is tied to North American Plate
- WGS84 is averaged over the world

→ They do not move in the same way

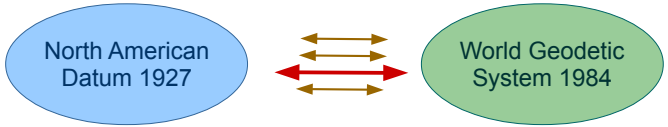
- 30 years ago, NAD83 ≈ WGS84
- Now, differ by about 1.5 metres

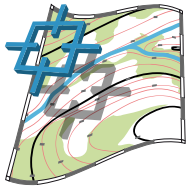



```
CoordinateOperation op = CRS.findOperation(source, target, aoi);
```

Coordinate Operation inspected

```
System.out.println(op);
```





Well Known Text (WKT) *version 2*

```
COORDINATEOPERATION[ "NAD27 to WGS 84 (3)",
  SOURCECRS[ GEODETICCRS[ "NAD27",
    DATUM["North American Datum 1927",
      ELLIPSOID["Clarke 1866", 6378206.4, 294.9786982138982]],
    CS[ellipsoidal, 2],
      AXIS["Latitude (Lat)", north],
      AXIS["Longitude (Lon)", east],
      ANGLEUNIT["degree", 0.01745329251994329 5],
    ID["EPSG", 4267, "8.9"]],
  TARGETCRS[… definition omitted for brevity …],
  METHOD["Geocentric translations (geog2D domain)"],
    PARAMETER["X-axis translation", -10, UNIT["metre", 1]],
    PARAMETER["Y-axis translation", 158, UNIT["metre", 1]],
    PARAMETER["Z-axis translation", 187, UNIT["metre", 1]],
  OPERATIONACCURACY[20],
  SCOPE["Accuracy 15m, 11m and 6m in X, Y and Z axes."],
  AREA["Canada - onshore and offshore."],
  BBOX[40.04, -141.01, 86.46, -47.74],
  ID["EPSG", 1172, "8.9",
    URI["urn:ogc:def:coordinateOperation:EPSG:8.9:1172"]],
  REMARK["Derived at 112 stations."]]
```

… map projection part omitted …

Coordinate operation applied

```
MathTransform mt = op.getMathTransform();
```

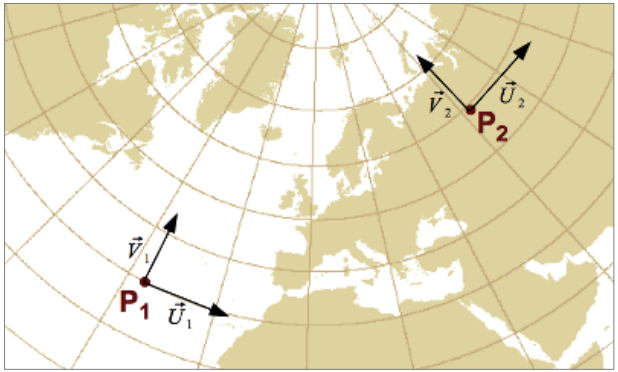
Example for a two-dimensional map projection:
(number of rows or columns depend on the number of dimensions)

`mt.transform(φ, λ)` :

$$\begin{pmatrix} x \\ y \end{pmatrix}$$

`mt.derivative(φ, λ)` :

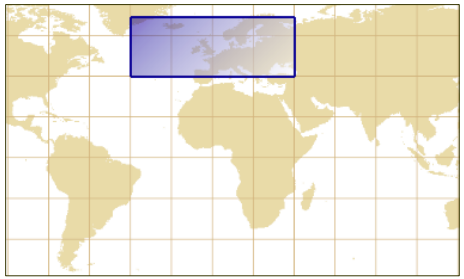
$$\begin{pmatrix} \partial x/\partial\varphi & \partial x/\partial\lambda \\ \partial y/\partial\varphi & \partial y/\partial\lambda \end{pmatrix}$$

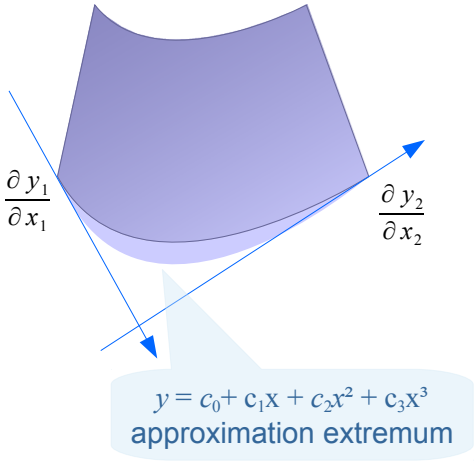


Coordinate operation on envelope

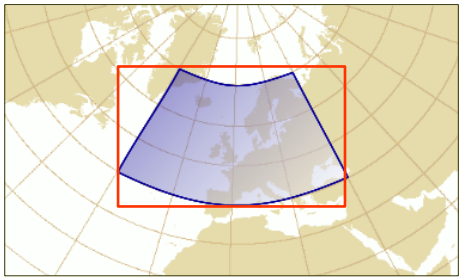
```
Envelope transformed = Envelopes.transform(op, envelope);
```

Source





Target



Also have special handling for envelopes over a pole

Use with planetary bodies

- Geodetic versus geocentric latitude
- Where is east or west?
- Prime meridian definition
- [0° … 360°] longitude range
- Handling high ellipsoid flattening
- Triaxial ellipsoid

Geodetic versus geocentric latitude

- Geographic systems use geodetic latitudes
- Planetary systems use both

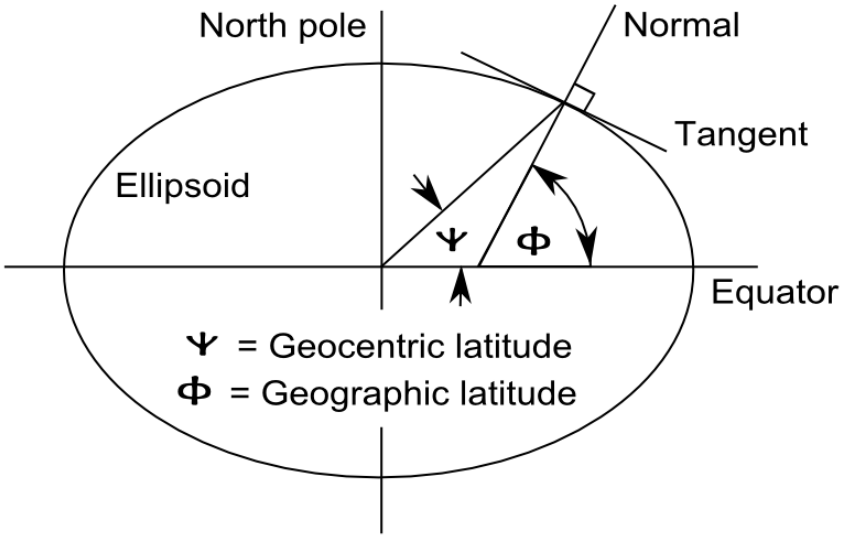


Image by Wmc824 on Wikimedia

Geodetic latitude

```
GEODETICCRS["Mars",
  DATUM["Mars",
    ELLIPSOID["Mars reference", TBD]],
  CS[ellipsoidal, 2],
    AXIS["Geodetic latitude (Lat)", north],
    AXIS["Geodetic longitude (Lon)", east],
    UNIT["degree", 0.017453292519943295]
```

Geocentric latitude

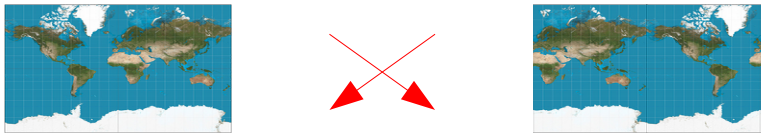
```
GEODETICCRS["Mars",
  DATUM["Mars",
    ELLIPSOID["Mars reference", TBD]],
  CS[spherical, 3],
    AXIS["Geocentric latitude (Lat)", north, UNIT[…]],
    AXIS["Geocentric longitude (Lon)", east, UNIT[…]],
    AXIS["Radial distance (R)", up, UNIT[“metre”, 1]]]
```

Longitude

- Definition of East and West directions
  - Depend on direction of planet rotation
  - Earth and Sun have different definition
- Definition of prime meridian
  - Some software have “Greenwich” hard-coded
- Longitude range
  - Most terrestrial systems use [-180° … 180°]
  - Planetary systems use [0° … 360°]

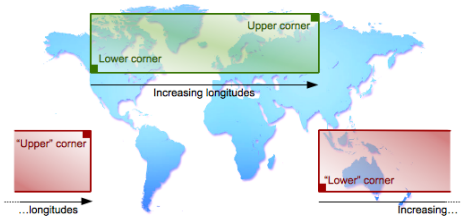
Longitude range implications

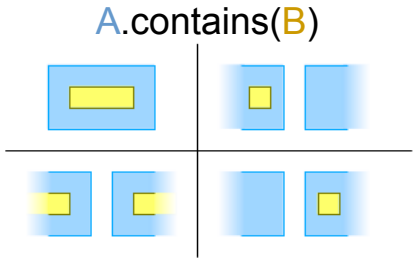
- Trigonometric functions targeting [-π … +π] range
- Need to interchange hemispheres in raster data



```
asin
acos
atan
```

- Complicate union / intersect / contain operations on envelopes





- Partial SIS support:

```
crs = crs.forConvention(AxesConvention.POSITIVE_RANGE);
```

High ellipsoid flattening (1/2)

- Some map projection formulas have no exact solution
- Two approximations:

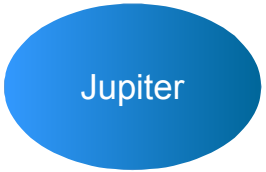
1) Iterative calculation:  adaptive but relatively slow.

2) Series expansion: faster, but the number of terms is designed for Earth.

$$\chi + C_1 \sin(2\chi) + C_2 \sin(4\chi) + C_3 \sin(6\chi) + C_4 \sin(8\chi)$$

$$+ C_5 \sin(10\chi) + C_6 \sin(12\chi) + \ldots$$

Enough terms for a planet flattened like Earth.
Map projection formulas typically stop here.



More terms needed for
more flattened planets.

High ellipsoid flattening (2/2)

- Software typically use one of the two approach

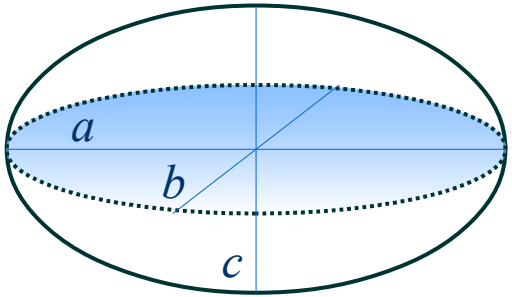
  ⚠ When using series expansion, there is usually no warning if accuracy is not sufficient

- Apache SIS uses an hybrid approach:
  - Begin with series expansion
  - If accuracy is not sufficient, continue with iterative method
  - Implemented for Mercator, Lamber Conic Conformal, Albers Equal Area, Cylindrical Equal Area

$$\chi + C_1 \sin(2\chi) + C_2 \sin(4\chi) + C_3 \sin(6\chi) + C_4 \sin(8\chi)$$ +

Stop here for map projections on Earth | Planetary projections

Triaxial ellipsoid



WKT 2 extension

```
GEODETICCRS["Io",
  DATUM["Io 2009",
    TRIAXIAL["Io 2009 IAU IAG", 1829400, 1819400, 1815700]],
  CS[ellipsoidal, 2],
    AXIS["Latitude (Lat)", north],
    AXIS["Longitude (Lon)", east],
    UNIT["degree", 0.017453292519943295]
```

- Cited in ISO 19111, but not yet standardized
- Require adapting map projection methods
- Not yet implemented in Apache SIS (exploring only)

Conclusion

Apache SIS advantages

- Rich geospatial metadata model from ISO standards
- Data discovery support for different data sources
- XML bindings hide some GML complexity
- Well Known Text (WKT) version 1 and 2
- Direct connection to an EPSG database
- “Late binding” referencing engine
- Coordinate operations use function derivatives for accuracy
- Extended precision arithmetic in critical parts
- Run Geospatial Integrity of Geoscience Software (GIGS) tests
- Support for high eccentricity in some map projections
- Extensive Javadoc

Future developments

- Upgrade metadata XML schema to ISO 19115-3
- Better vertical coordinate transformations
- Dynamic datum (following ISO 19111 revision)
- Support for grid coverage data (rasters)
- Filters for geospatial queries
- Catalog Service on the Web (based on Google Summer of Code 2016)

Longer term:

- Geometries
- Renderer
- OGC web services