Requirements for OAIS Structure Representation Information

Stephen Rankin (STFC), Matt Dunckley (STFC), Brian Mcilwrath (STFC), Esther Conway (STFC), David Giaretta (STFC), CASPAR and DCC

What is Structure RepInfo?

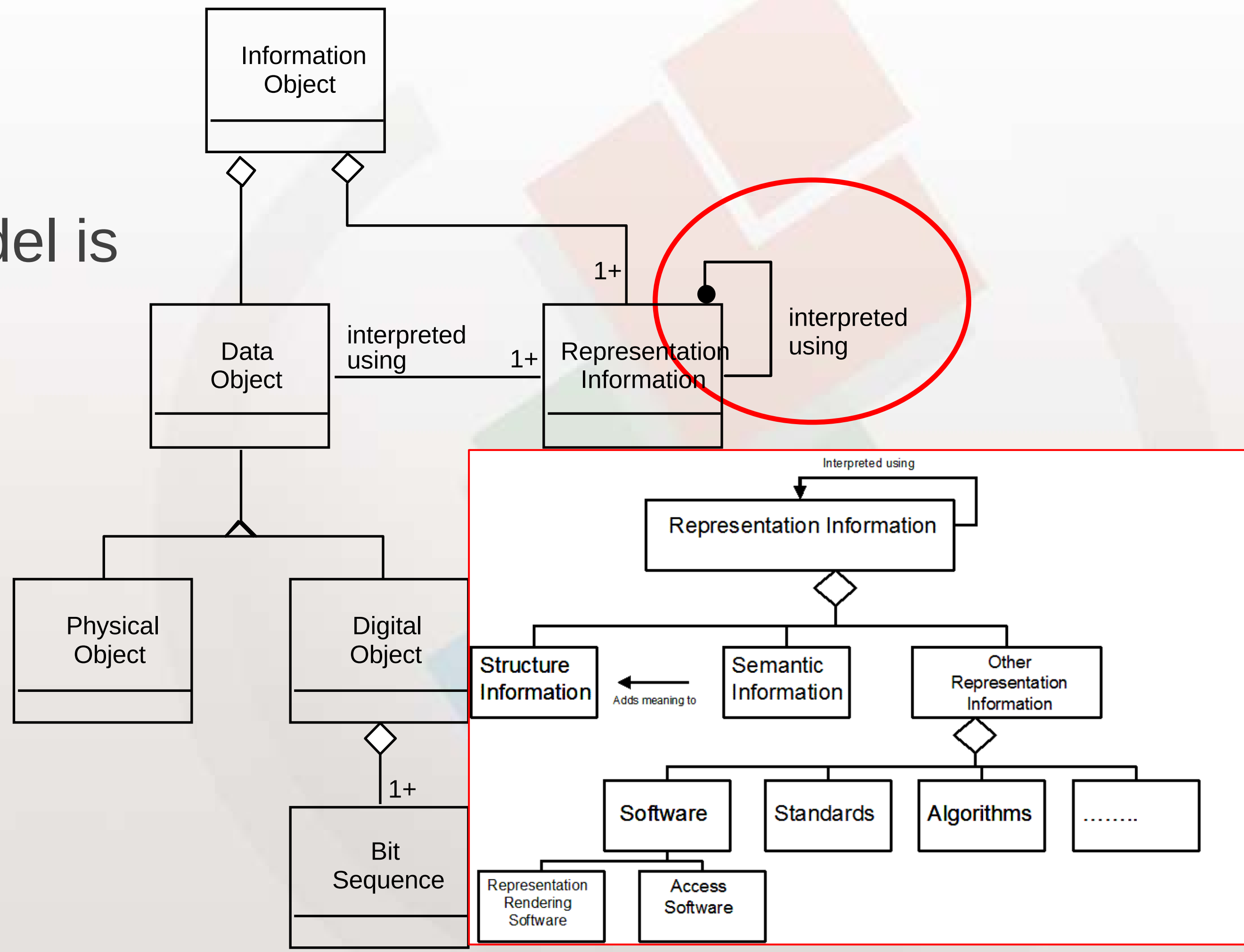
- OAIS definition:
  - *"Structure Information: The information that imparts meaning about how other information is organized. For example, it maps bit streams to common computer types such as characters, numbers, and pixels and aggregations of those types such as character strings and arrays."*
  - *"The Digital Object, as shown in figure 4-10, is itself composed of one or more bit sequences. The purpose of the Representation Information object is to convert the bit sequences into more meaningful information. It does this by describing the format, or data structure concepts, which are to be applied to the bit sequences and that in turn result in more meaningful values such as characters, numbers, pixels, arrays, tables, etc. These common computer data types, aggregations of these data types, and mapping rules which map from the underlying data types to the higher level concepts needed to understand the Digital Object are referred to as the Structure Information of the Representation Information object. These structures are commonly identified by name or by relative position within the associated bit sequences."*

Information Model & Representation Information

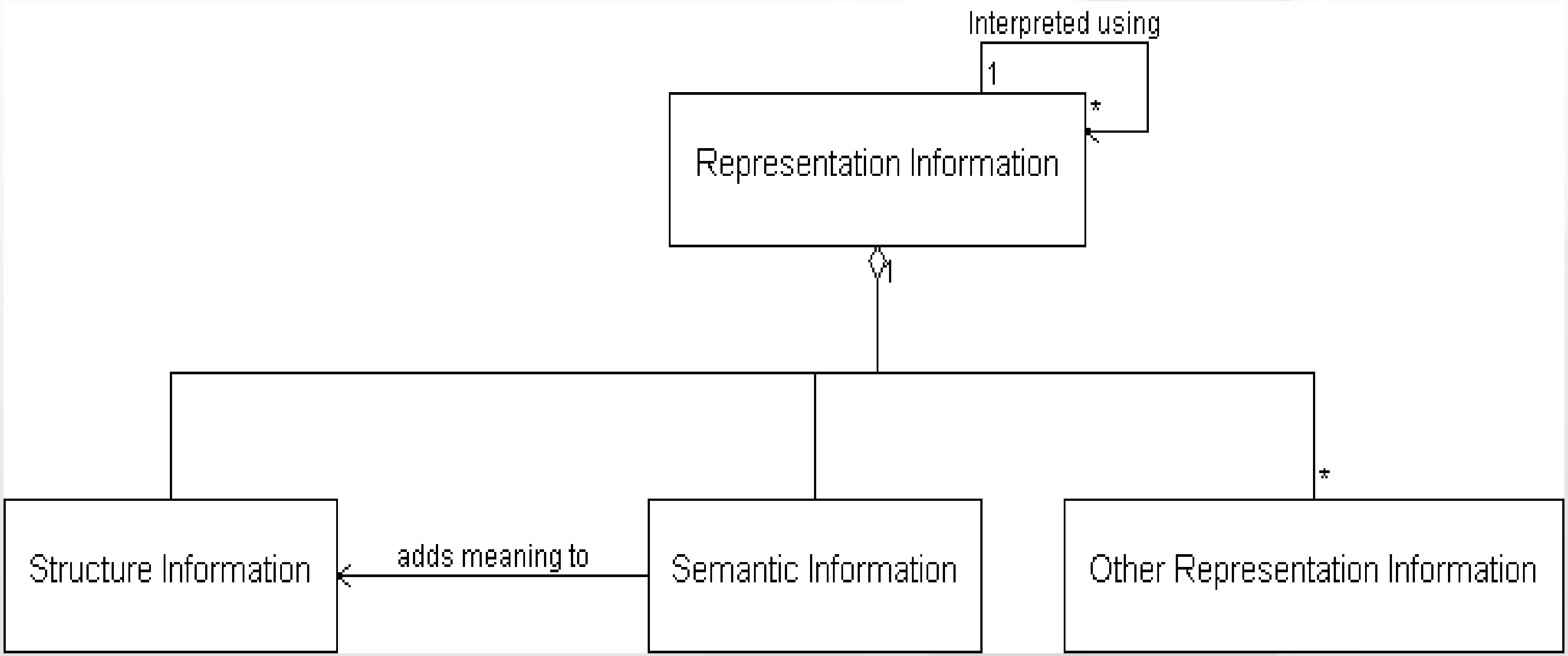
The Information Model is key

Recursion ends at KNOWLEDGEBASE of the DESIGNATED COMMUNITY

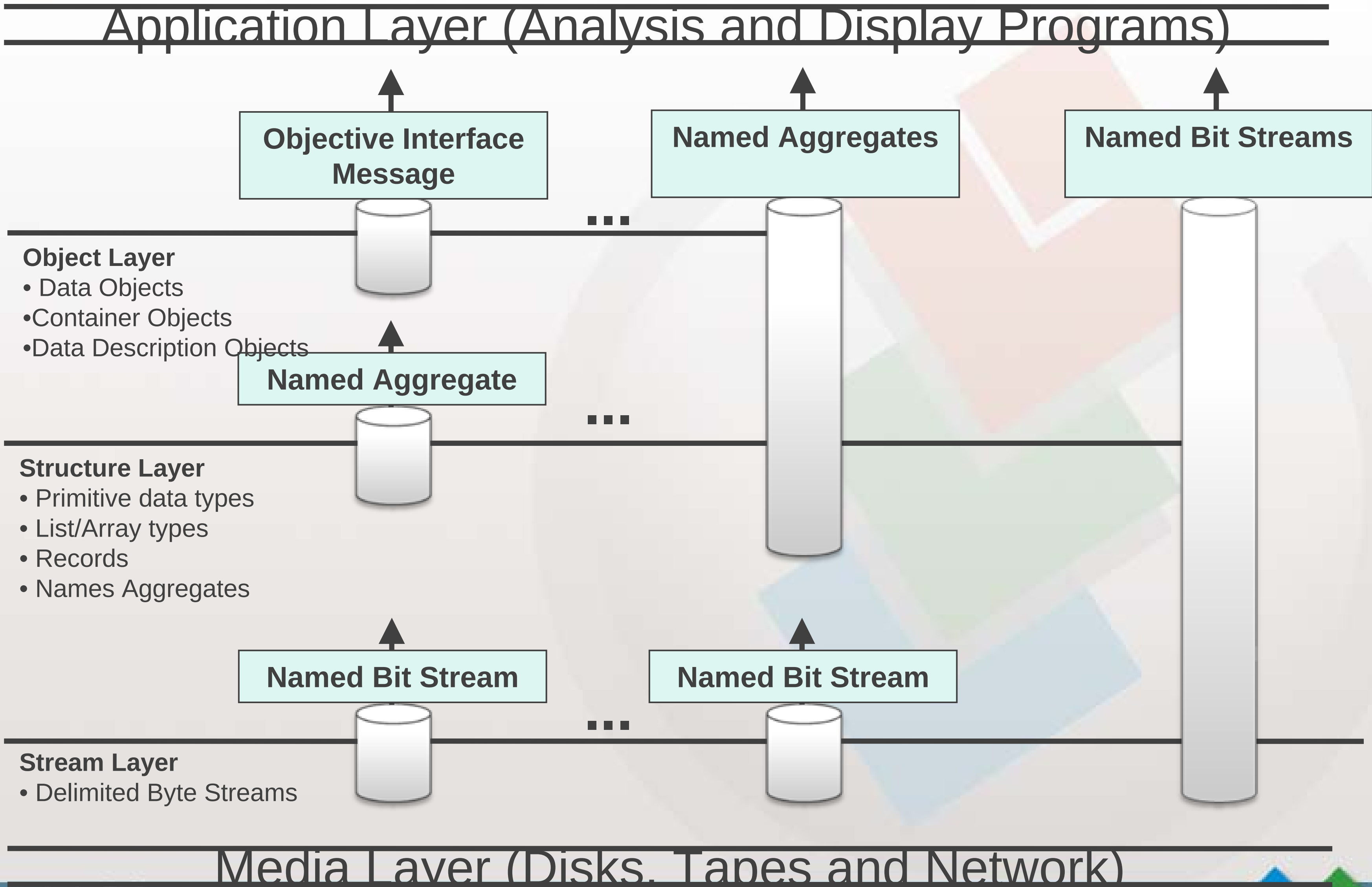
(this knowledge will change over time and region)



OAIS Representation Information and Representation Information Networks



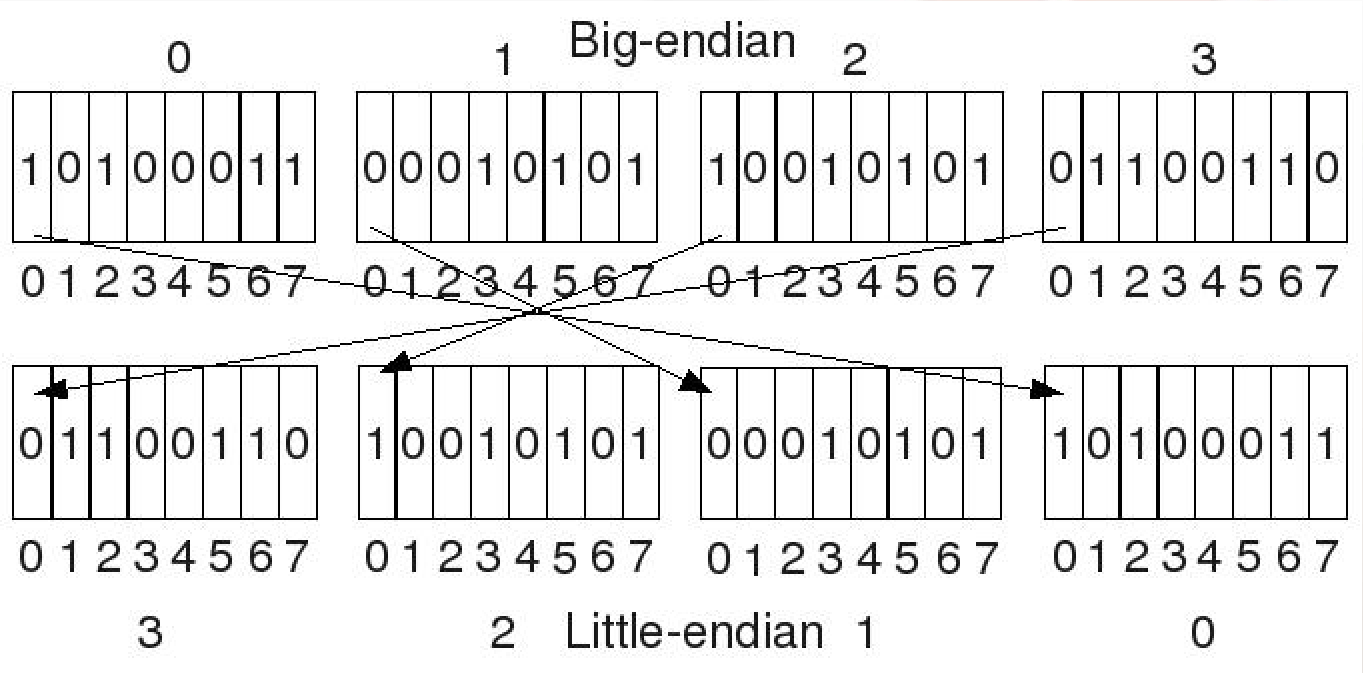
Formation layers



The Bits

- Each digital data object is composed of a sequence of bits, which are simply zeros or ones.
- Bits are usually grouped together to encode and represent some form of data value.
- Integer, Character, String, Boolean, Real Floating Point, Enumeration, Marker, Record or Custom

Bit Order



Big-endian
0
1
2
3
1 0 1 0 0 0 1 1
0 0 0 1 0 1 0 1
1 0 0 1 0 1 0 1
0 1 1 0 0 1 1 0
0 1 2 3 4 5 6 7
0 1 1 0 0 1 1 0
1 0 0 1 0 1 0 1
0 0 0 1 0 1 0 1
1 0 1 0 0 0 1 1
0 1 2 3 4 5 6 7
3
2
1
0
Little-endian

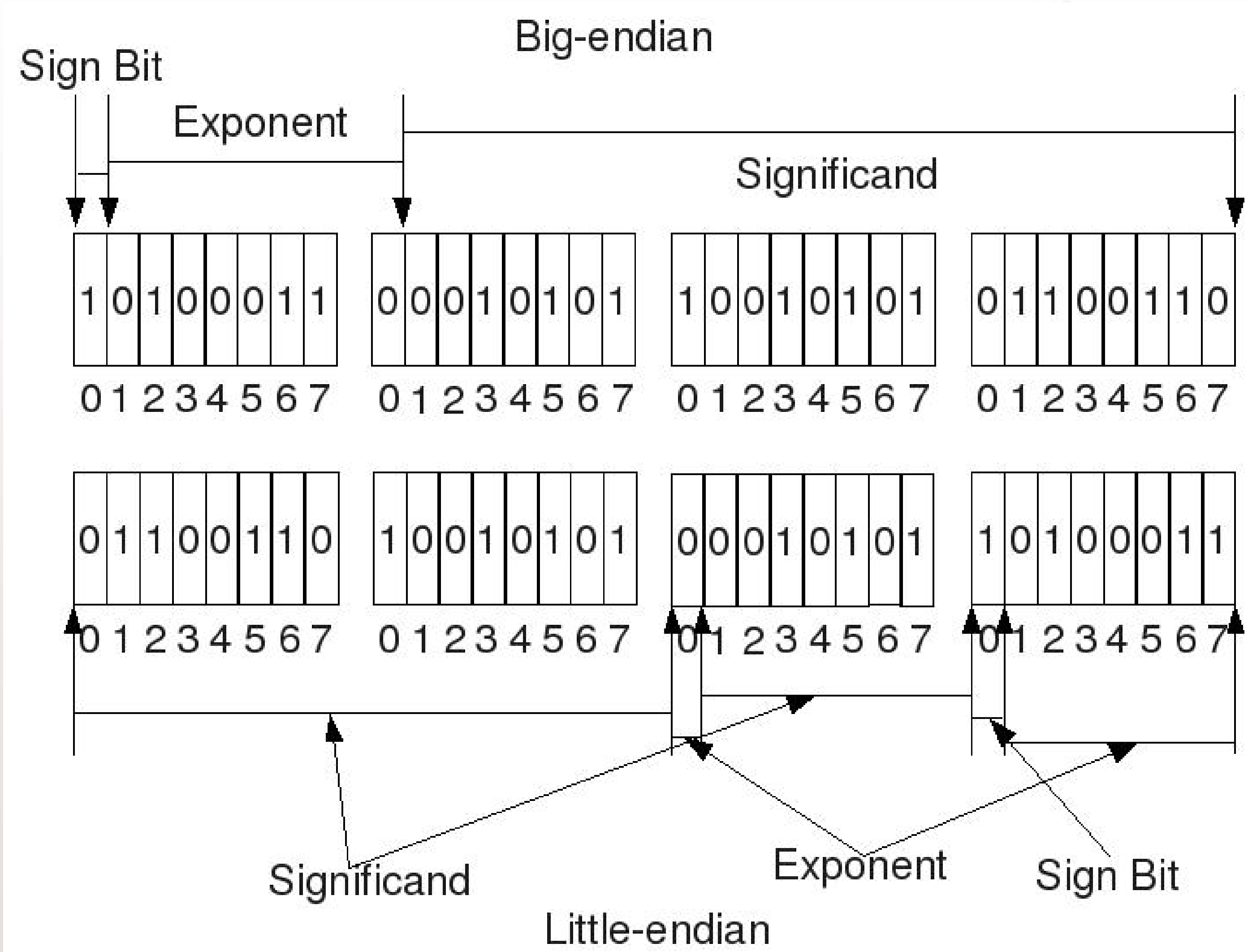
Integer Properties

- Endienness.
- Size in octets/bits.
- Signed/unsigned.
- Location of signed bit.
- Interpretation method - two's compliment Sign and Magnitude, One’s compliment etc.
- Restriction on maximum and minimum size, fixed number of values.

Character Properties

- Character set used.
- Size in octets/bits.
- Endienness.

Real Floating Point Properties



Real Floating Point Properties

- Endienness.
- Location and structure of the significand bits.
- Location and structure of the exponent bits.
- Normalised.
- Interpretation method of significand - two's compliment etc.
- Bias scheme for exponent.
- Reserve values/exceptions.
- Location of signed bit.
- Formula for interpreting the number.
- Restriction on maximum and minimum size. ,fixed number of values.

String Properties

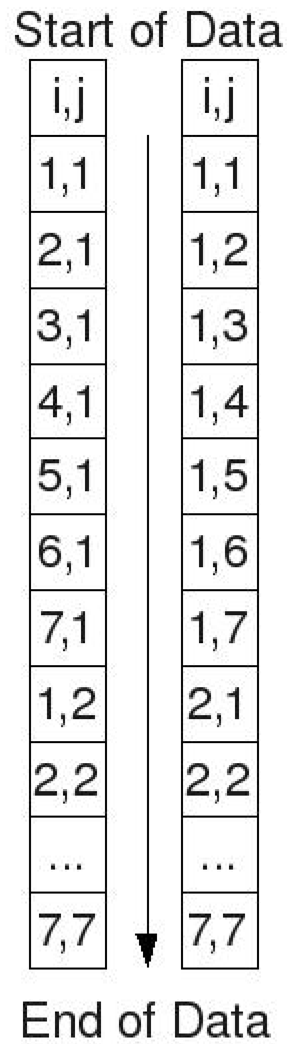
- Character set used.
- Size in octets/bits of each character.
- Structured or unstructured.
- If structured then a description of the structure such as BNF etc.
- The length in characters of the string.
- Expression for calculating the length of the string.
- Allowed characters in the string.
- Fixed values of strings.

Arrays

- Number of dimensions if static.
- Calculation of Number of dimensions if dynamic.
- Number of values in each dimension if static.
- Calculation of number of values in each dimensions if dynamic.
- Ordering of the arrays (row order or column order).
- Data type (integer, real etc).
- Restriction on maximum and minimum number of dimensions.
- Fixed number of values the dimensions of the array can take.
- Restriction on maximum and minimum number of values in a dimension.
- Fixed number for size of the dimensions of the array.
- Restriction on maximum and minimum values the values of the array can take.
- Markers indicating the end of a dimension or an array.

Arrays

i,j	i=1	i=2	i=3	i=4	i=5	i=6	i=7
j=1	1,1	2,1	3,1	4,1	5,1	6,1	7,1
j=2	1,2	2,2	3,2	4,2	5,2	6,2	7,2
j=3	1,3	2,3	3,3	4,3	5,3	6,3	7,3
j=4	1,4	2,4	3,4	4,4	5,4	6,4	7,4
j=5	1,5	2,5	3,5	4,5	5,5	6,5	7,5
j=6	1,6	2,6	3,6	4,6	5,6	6,6	7,6
j=7	1,7	2,7	3,7	4,7	5,7	6,7	7,7



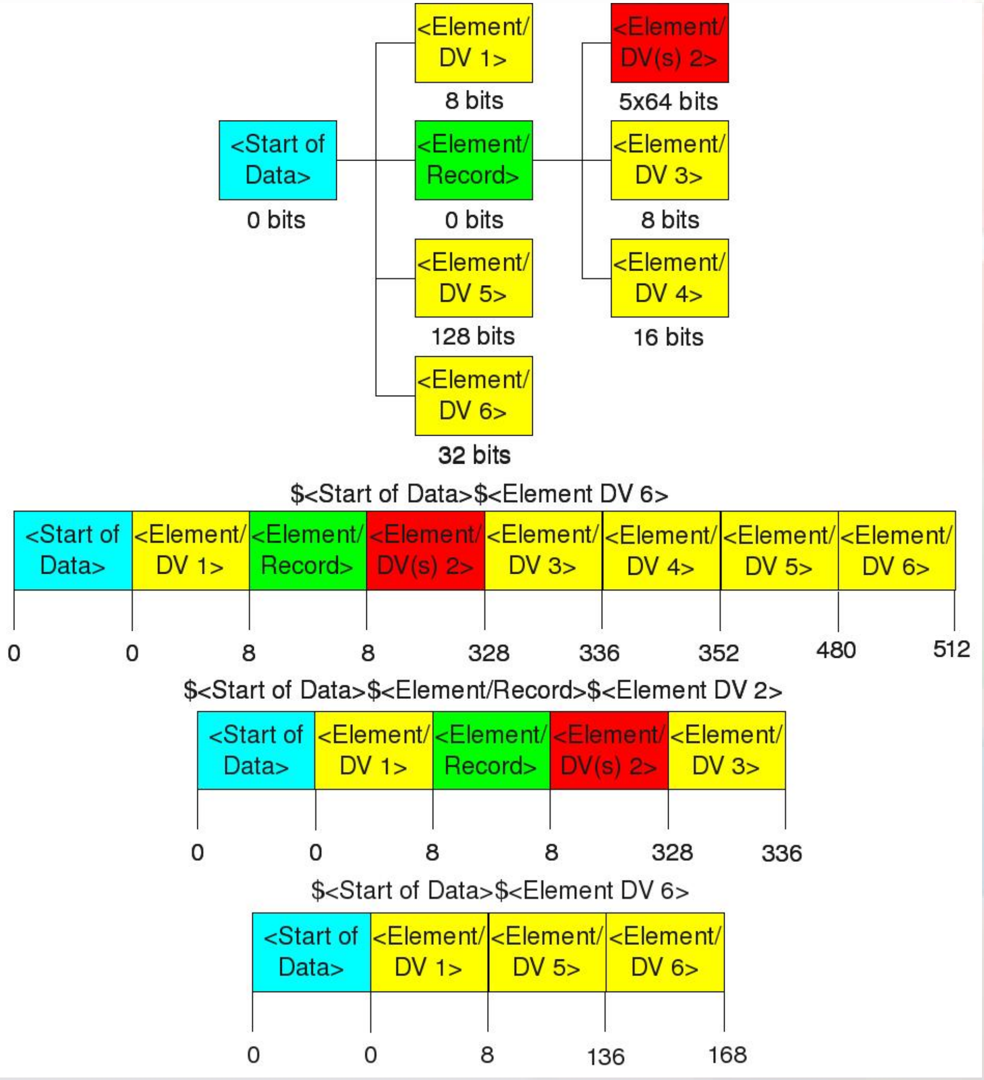
Custom Data Types

- Data can be manipulated at the bit level within software.
- Users can create their own data types.
- Bit packing used to be done to save space.

Other Data Types

- Boolean
  - Data type used to represent Boolean value.
  - Values of data type that represent true/false.
- Markers
  - Data type.
  - Values of the marker.
- Enumerations
  - Data types of enumeration.
  - Number of enumeration values.
  - The enumeration table.
- Records
  - Existence expression.
  - Child elements and their order.
  - Parent element.

Logical Structure



Logical Structure

- Elements and their names.
- Element Primitive Data Types.
- Path statements with predicates for accessing array elements.
- Offset values and calculation for offsets from other DVs.
- Calculation of the existence of elements or records from other DVs in a logical expression.
- Logical - Comparison expressions, i.e. string comparisons etc. Existence values. Choice statements of elements or records.

Why and how?

- Capture enough information to automatically, using only the formal descriptions, load data into new applications.
- Abstraction of required information (API).
- Validation – generate access code and load data into new application.

Formal Descriptions of Semantics

DEDSL Abstract, PVL, and XML(DTD) syntax for defining some simple data semantics.

Only a small number of required attributes for a given data structure, NAME, DEFINITION, UNITS (conditional), *ENTITY*_TYPE (conditional), ENUMERATION_VALUES (conditional), TEXT_SIZE (conditional).

You can define your own attributes.

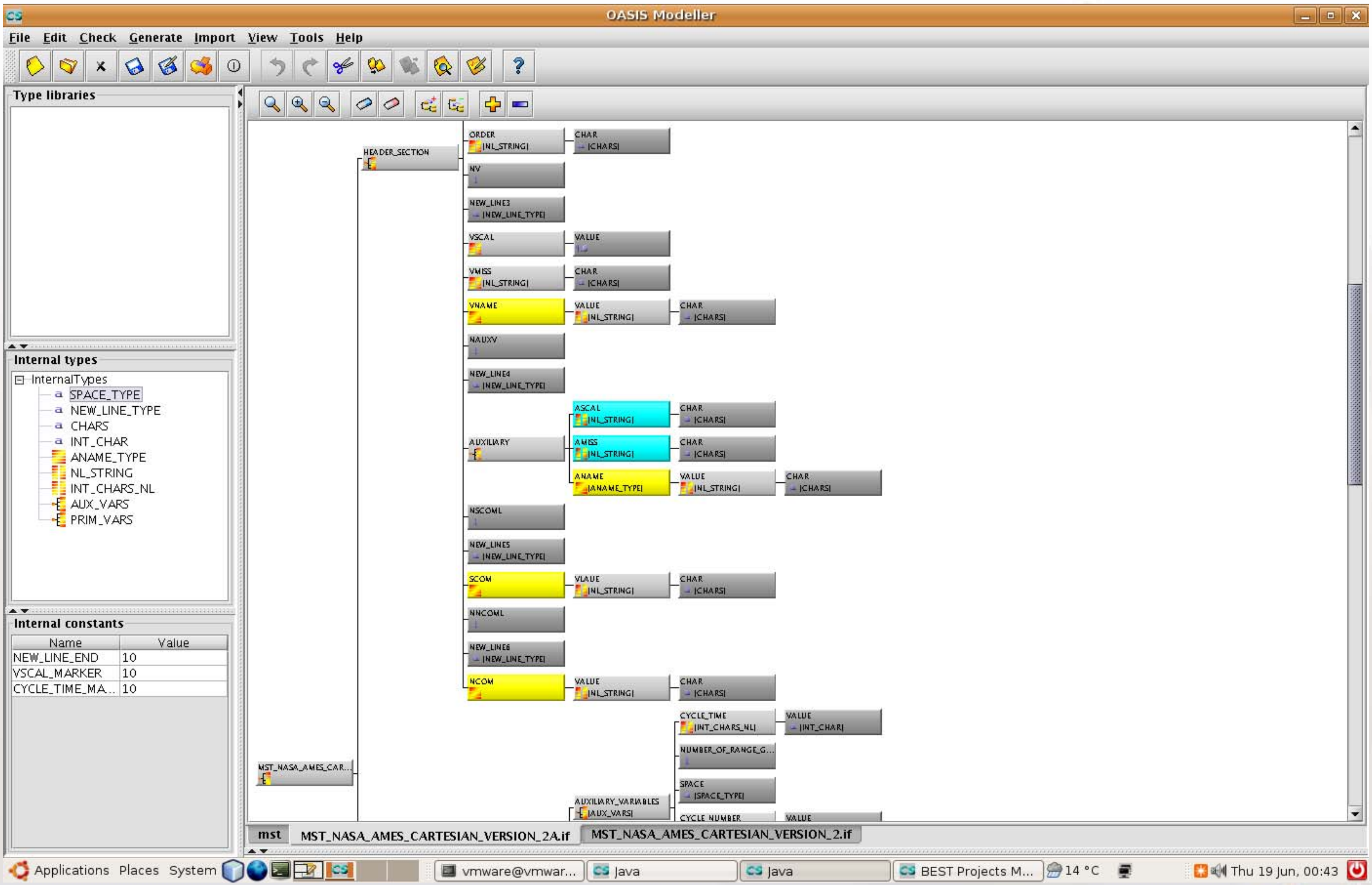
You can reuse definitions from other dictionaries.

Link the data structures to the semantics via the EAST access path or an XPATH, i.e. define a new attribute – EAST_PATH (OASIS tool does this).

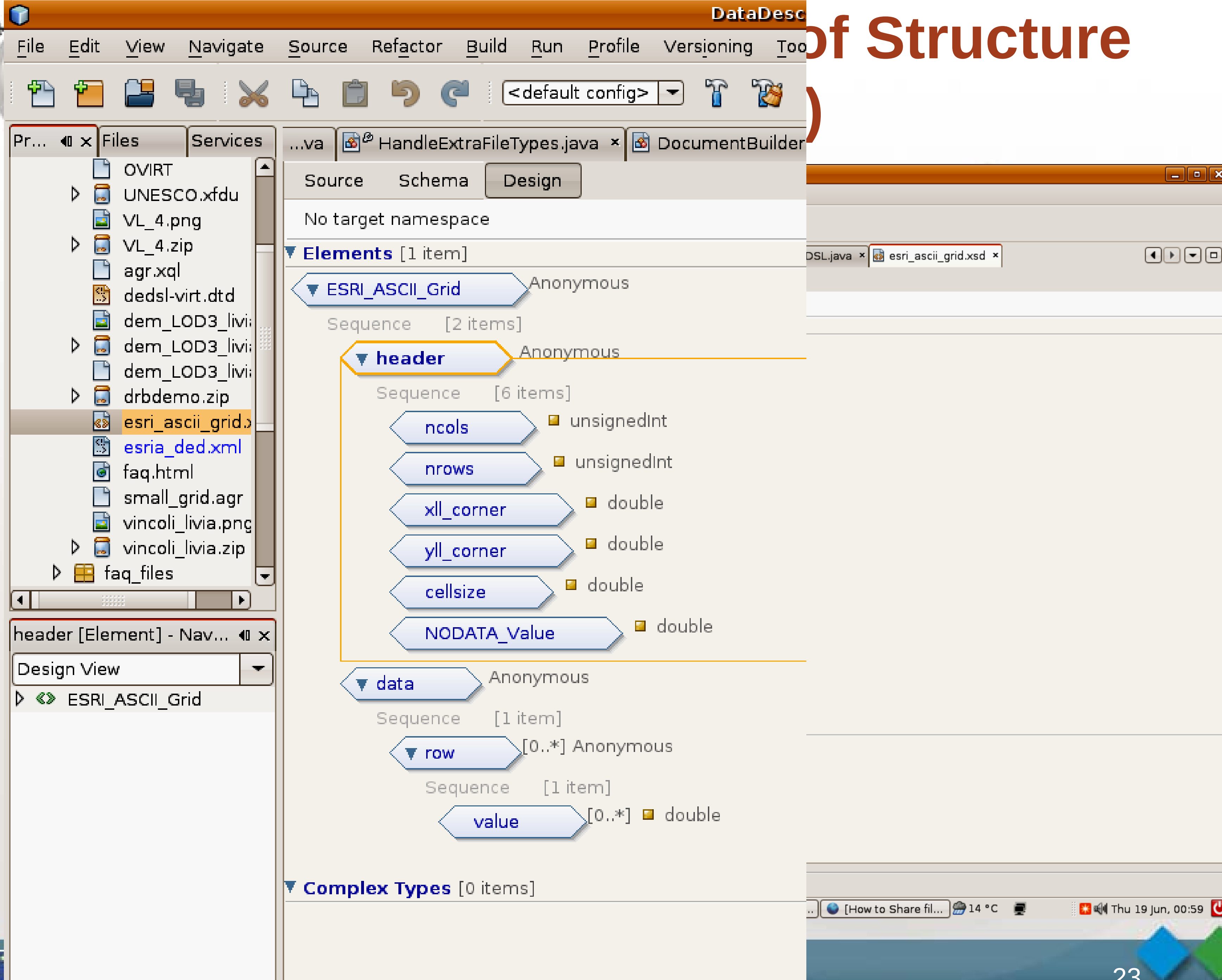
Formal Descriptions of Structure and Semantics

- CNES EAST tools (http://east.cnes.f), OASIS, EAST C Library (reference implementation).
- Also DEBAT (BEST Tools) http://debat.c-s.fr/
- Data Request Broker (DRB) -http://www.gael.fr/drb/site/
- JNI Wrapper for EAST C Library in our SVN repository (jnieast).
- DEDSL Abstract, PVL, and XML(DTD) syntax for defining some simple data semantics. RDF, RDFS and OWL.
- Interfaces for a more general data description language and semantics API (on Sourceforge SVN) (DSSIL).
- GUI Tools for capturing Object Oriented Semantics (RDF and RDFS) and Code Generation.

Formal Descriptions of Structure (Examples)



# of Structure
)

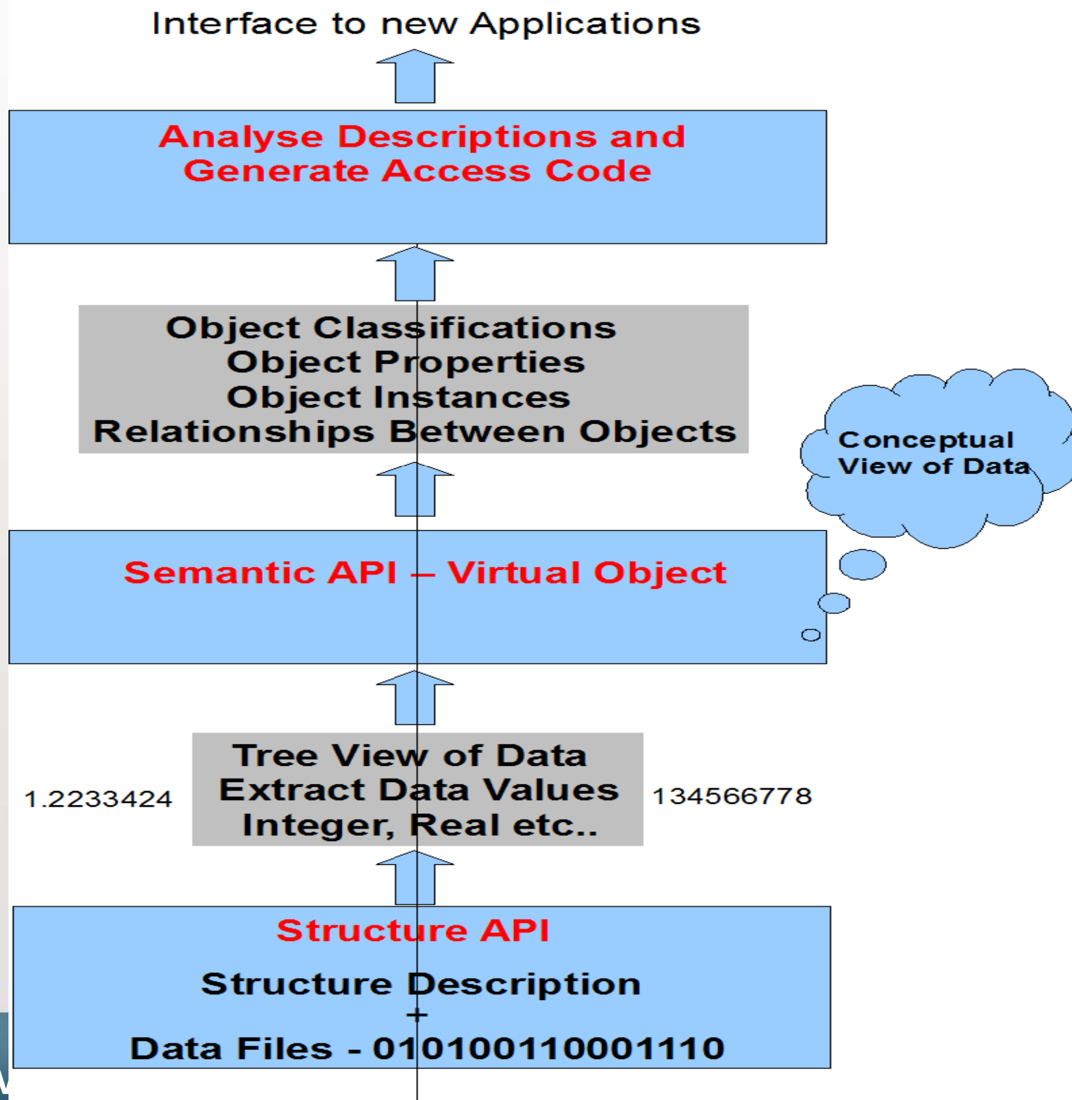


OSCAR Object Oriented Data Semantics

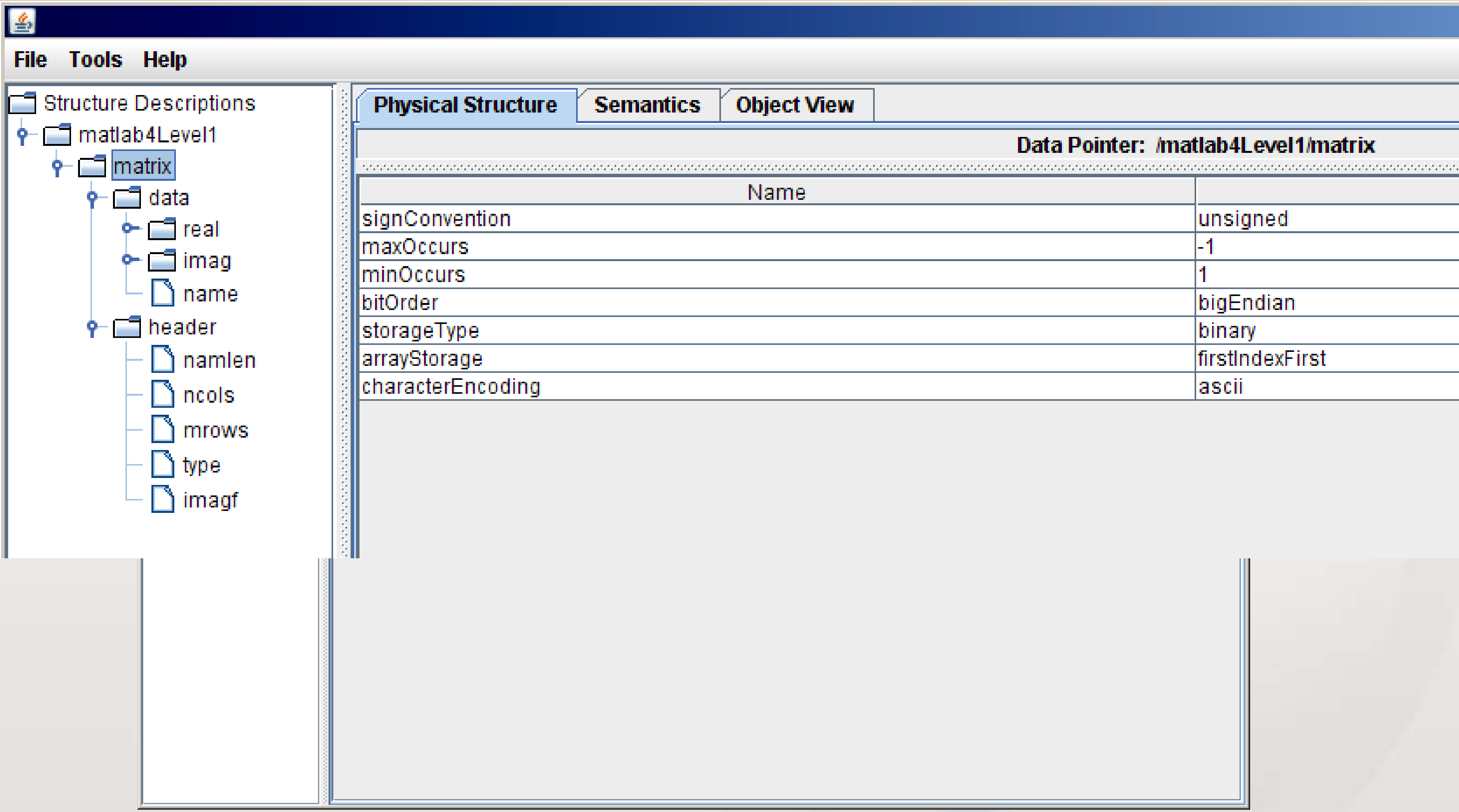
- Can create formal structure RepInfo using tools like DRB and EAST using tools in RepInfo Toolbox.
- DRB and EAST just allow you to get the data values and the basic type information (integer, real etc…) using a unique pointer to the values.
- OSCAR allow the addition of object oriented semantic RepInfo to the data.
- Analyses the structure and semantics to generate code that can be used to import data objects into new applications.
- Applications usually exist now that read the data, but this may not be true in the future. Future users will have to read data into new applications.

Basic Concept of OSCAR

Velocity m/s	Distance m
1.234234	23
1.435423	21
1.235453	20
1.235235	18



View Structure



Java™ 2 Platform Standard Ed. 5.0

javax.swing.table

Class AbstractTableModel

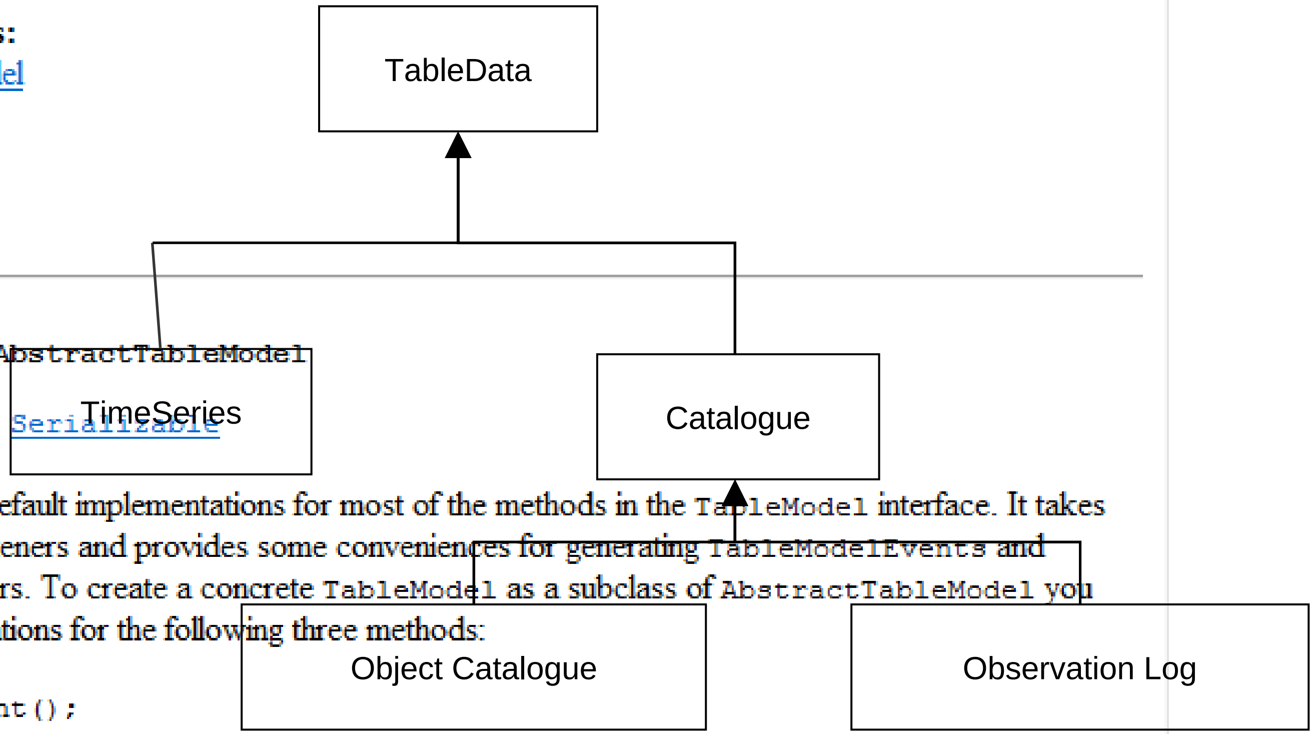
```
java.lang.Object
  └ javax.swing.table.AbstractTableModel
```

All Implemented Interfaces:

Serializable, TableModel

Direct Known Subclasses:

DefaultTableModel




```
public abstract class AbstractTableModel
extends Object
implements TableModel, Serializable
```

This abstract class provides default implementations for most of the methods in the `TableModel` interface. It takes care of the management of listeners and provides some conveniences for generating `TableModelEvents` and dispatching them to the listeners. To create a concrete `TableModel` as a subclass of `AbstractTableModel` you need only provide implementations for the following three methods:

```
public int getRowCount();
public int getColumnCount();
public Object getValueAt(int row, int column);
```

Add Object Oriented View

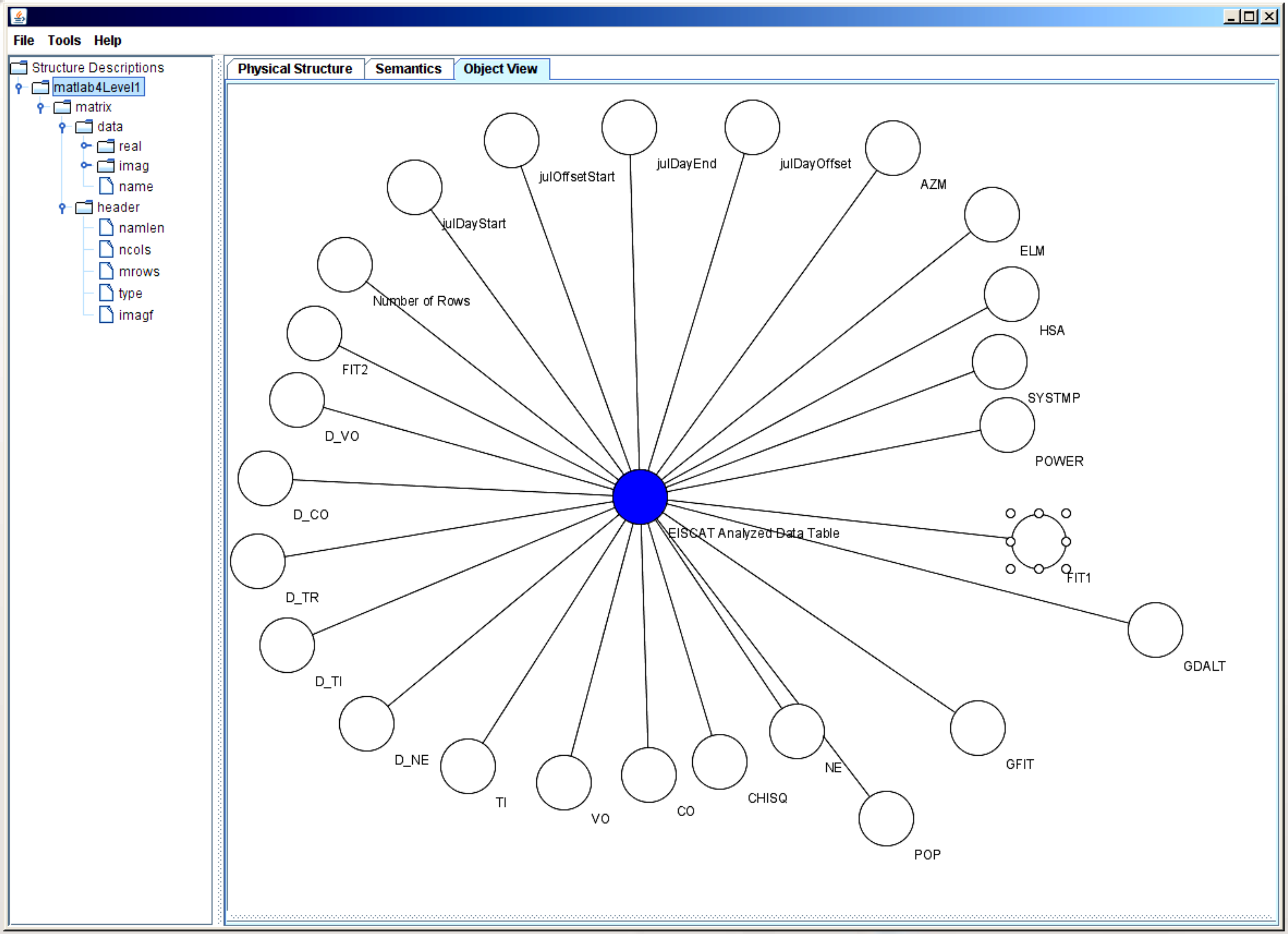
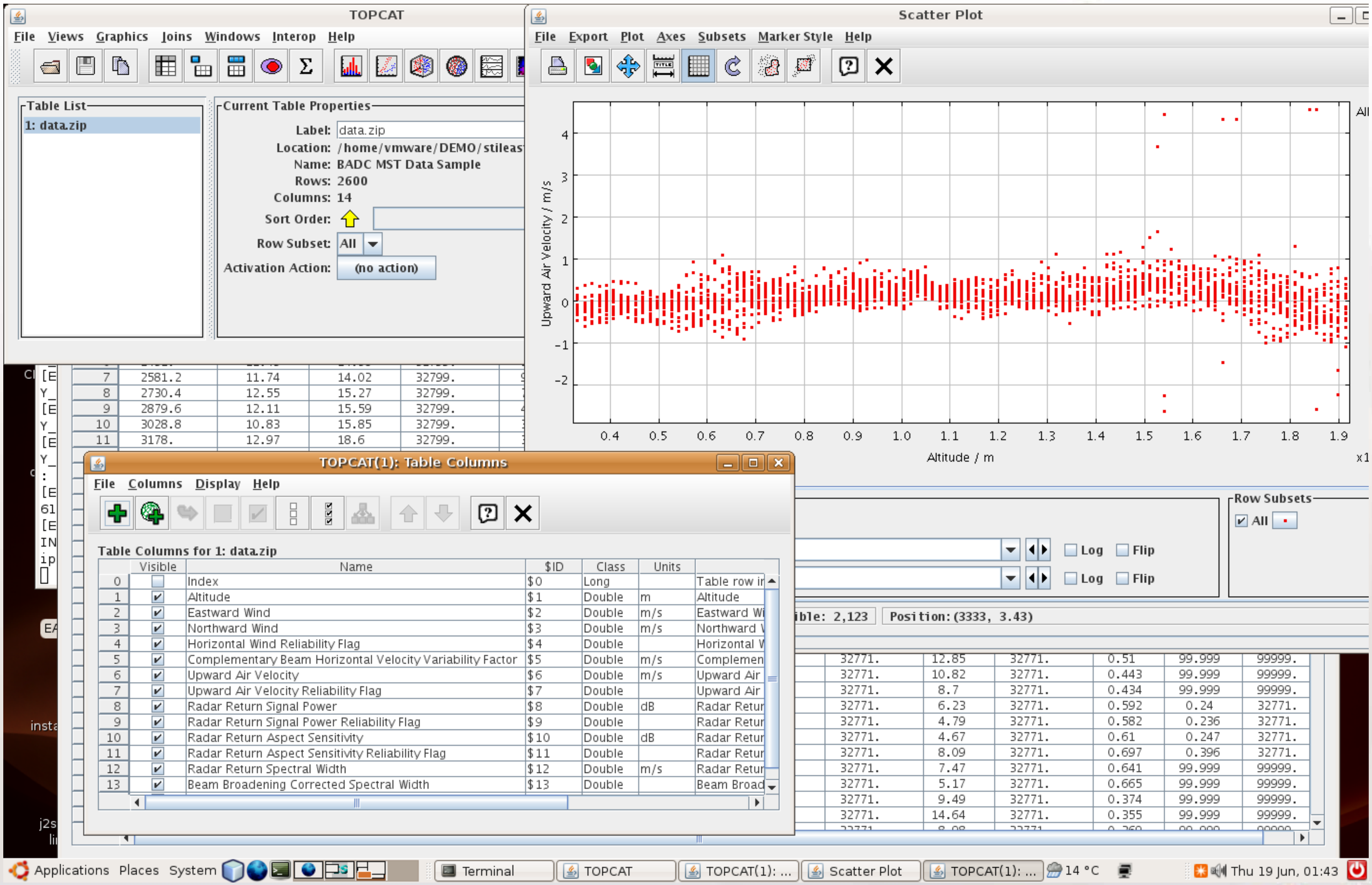


Table Data re-use example



Conclusion

- The details of the bits and how they map into data values can be complicated.
- For long-term preservation, the details of the bits is important for data reuse.
- Logical structure does not necessarily convey any meaning – may not relate to the type of object. Table, Image etc…
- But logical structure does allow the ordering of data values and calculation of their locations.
- Data pointers (paths) are important for addressing data values.
- Data re-use can be supported by appropriate data descriptions (Representation Information)

Links

- CASPAR – http://www.casparpreserves.eu
- DCC – http://www.dcc.ac.uk
- CASPAR videos - http://www.casparpreserves.eu/training/advanced-digital-preservation-training-lectures/
- CASPAR Source code - http://sourceforge.net/projects/digitalpreserve/ and http://developers.casparpreserves.eu:8080/hudson
  - jnieast - http://developers.casparpreserves.eu:8080/hudson/job/CASPAR-REPINF/ws/implementation/repinfotoolbox/jnieast/
  - DSSIL Interfaces – http://developers.casparpreserves.eu:8080/hudson/job/CASPAR-REPINF/ws/interfaces/repinfotoolbox/dssli/
  - DSSLI Implementations (partial) http://developers.casparpreserves.eu:8080/hudson/job/CASPAR-REPINF/ws/implementation/repinfotoolbox/dsslieast/ http://developers.casparpreserves.eu:8080/hudson/job/CASPAR-REPINF/ws/implementation/repinfotoolbox/dsslidrb/