XML information Packaging Standards for Archives

Lou Reich/CSC

Long Term Knowledge Retention Workshop
March15,2006

XML Packaging Standards

- Growing interest in XML-based representation of Information Object in Digital Library architectures:
  - Platform-independence,
  - Industry-support
  - Longevity, potential migration paths
  - Processing tools, validation capabilities
- XML-based Compound Object formats:
  - ISO/IEC 21000-2 MPEG-21 DID & DIDL
  - METS
  - IMS/CP
  - CCDS XFDU
- Typical functionality:
  - By-Value (base64) and/or By-Reference provision of constituent datastreams
  - By-Value and/or By-Reference provision of metadata
  - Provision of identifiers

From MPEG 1 to MPEG-21

- Potential impact of MPEG-21
  - MPEG: ISO/IEC Committee
  - MPEG-1/MPEG-2/MPEG-4/MPEG-7/MPEG-21/MPEG-A
  - Expected industry support
- 'MPEG-21 defines 'a normative open framework for multimedia delivery and consumption for use by all the players in the delivery and consumption chain'
  - Applicability to Digital Libraries
  - Ability to accomodate any media type and genre
- MPEG-21 is modular:
  - Part 2: DID – representation of digital objects
  - Part 3: DII – identification of digital objects
  - Part 4: IPMP – enforcement of rights expressions
  - Part 5: REL – declaration of rights expressions
  - Part 7: DIA – transcoding based on contextual information
  - Part 10: DIP – association of behaviors
  - Part 16: BF – binary representation of digital objects

MPEG-21 DID

- ISO/IEC 21000-2: Digital Item Declaration
  - First edition: ISO Standard – March 2004
  - Second edition: FDIS – January 2005 (based on LANL/UGent proposals related to DL use)
- Representation of Digital Objects
  - Within the MPEG-21 framework referred to as Digital Items
- 3 distinct sections:
  - Abstract Model – DID
  - Representation of the Model in XML (DIDL) – DIDL document
  - W3C XML Schema
- Other representation of DID may emerge:
  - ISO/IEC 21000-16: Binary Format
  - RDF?
- Reference software: ISO/IEC 21000-8

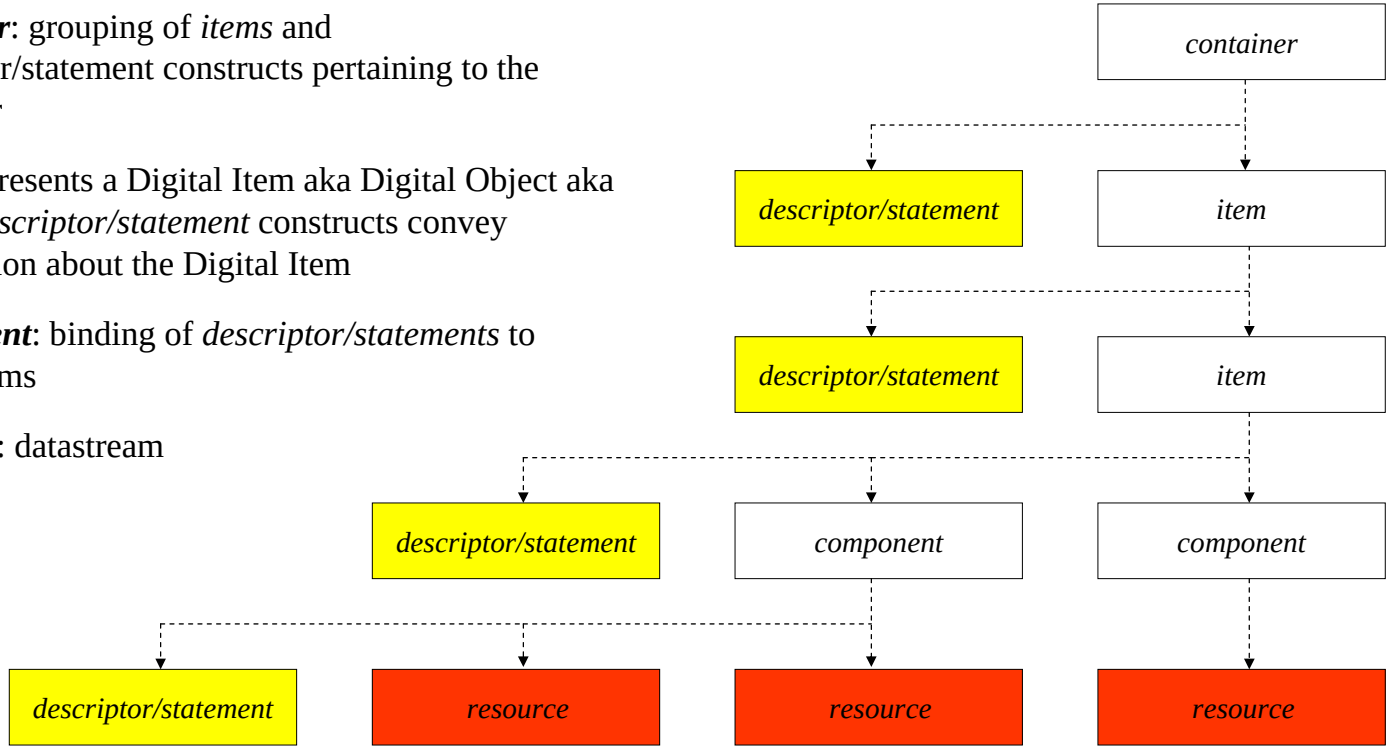
Abstract Model – basic entities-MPEG21

container: grouping of *items* and descriptor/statement constructs pertaining to the container

item: represents a Digital Item aka Digital Object aka asset. *Descriptor/statement* constructs convey information about the Digital Item

component: binding of *descriptor/statements* to datastreams

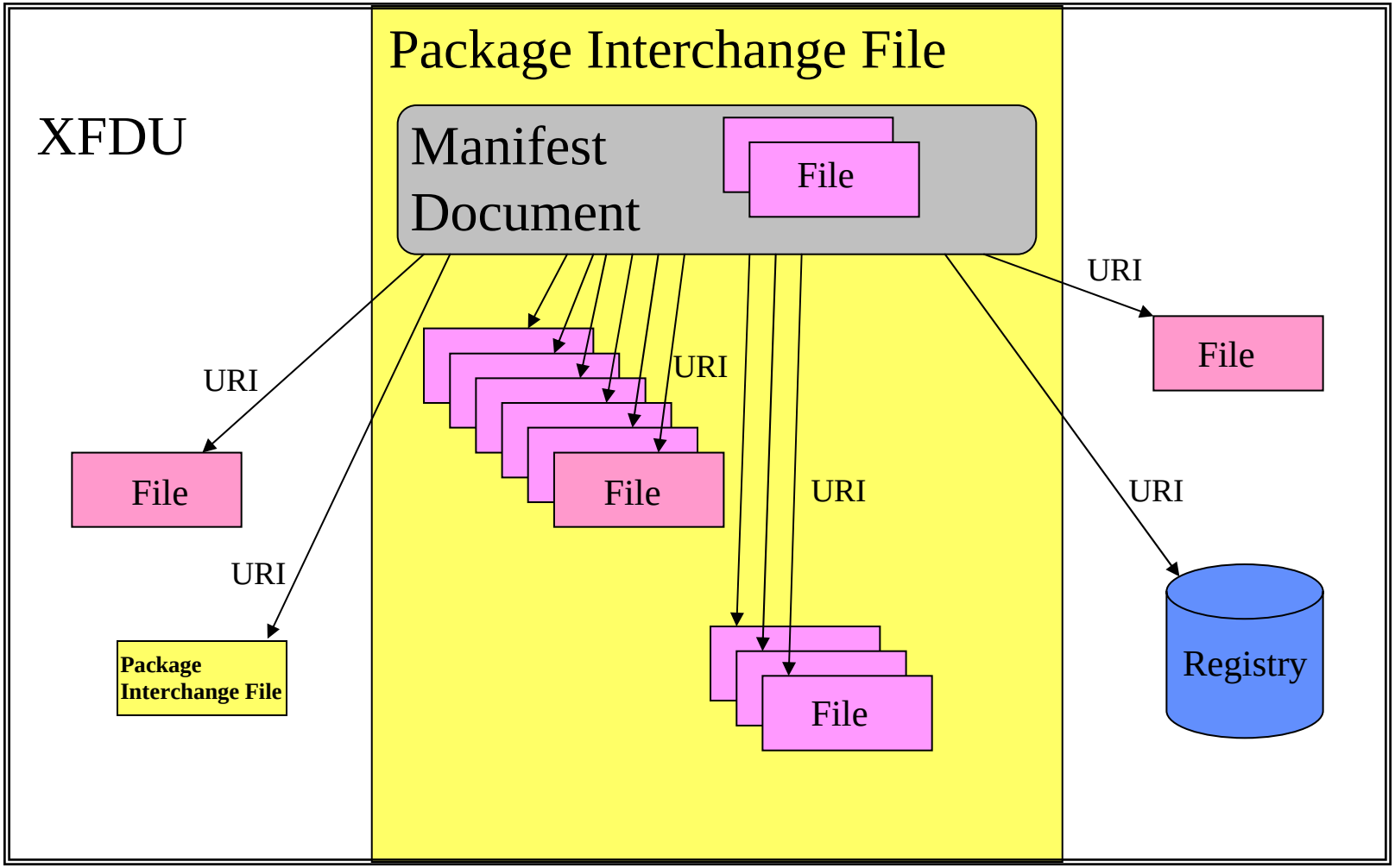
resource: datastream



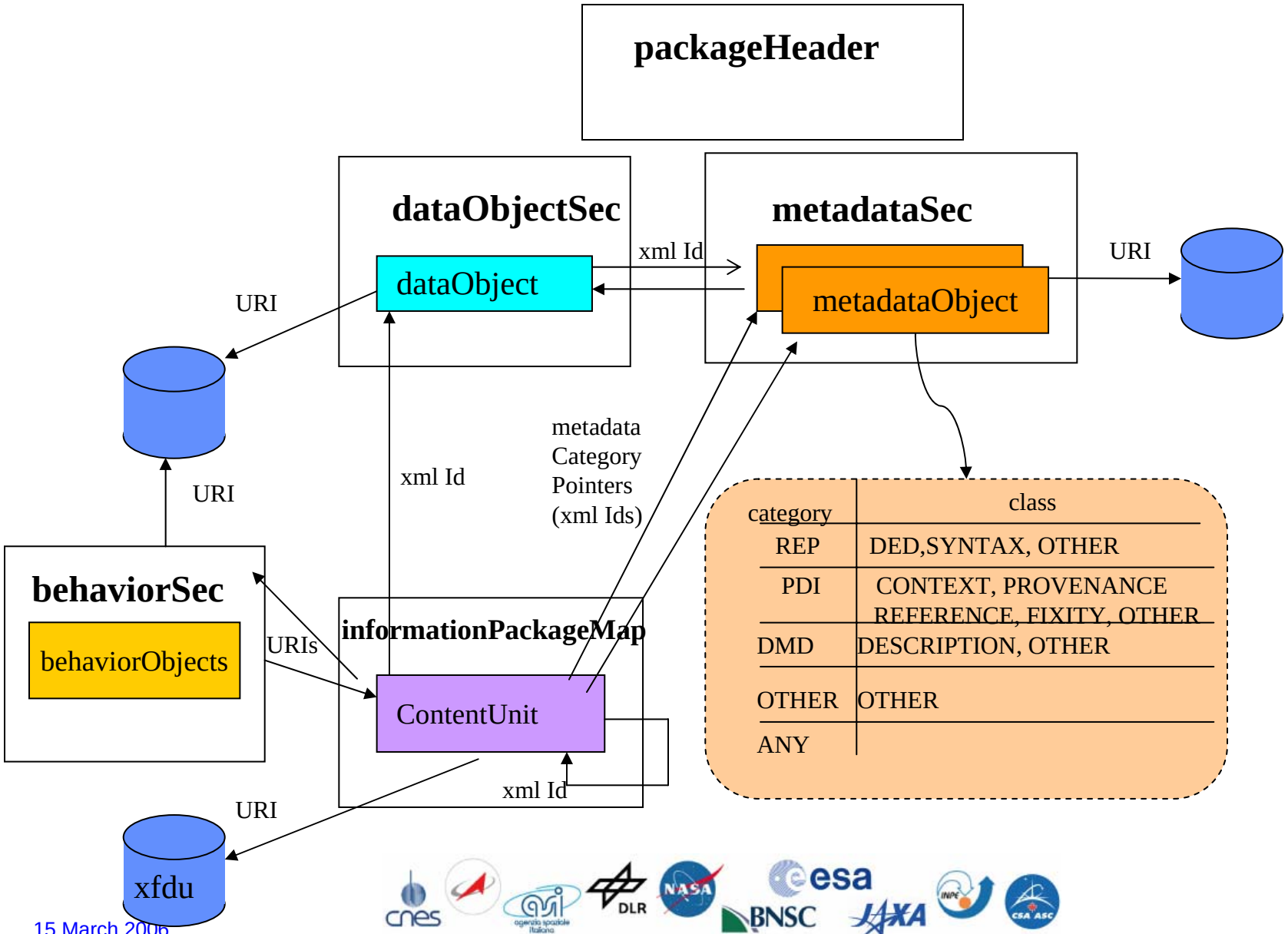
The METS Format

- Create a single document format for encoding digital library objects which can fulfill roles of SIP, AIP and DIP within the OAIS reference model
- Initial scope limited to objects comprised of text, image, audio & video files
- Promote interoperability of descriptive, administrative and technical metadata while supporting flexibility in local practice

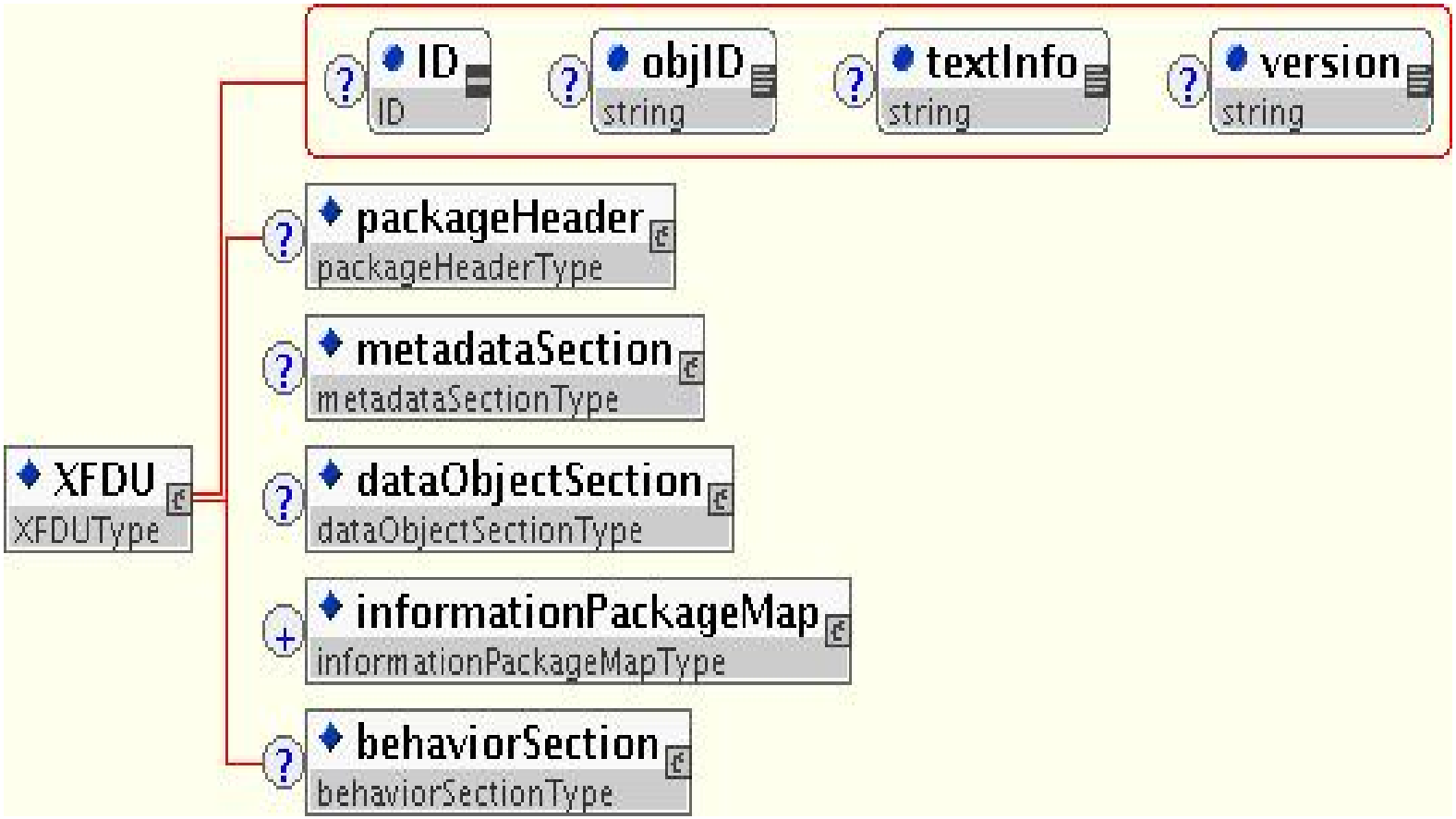
Information Package Concepts



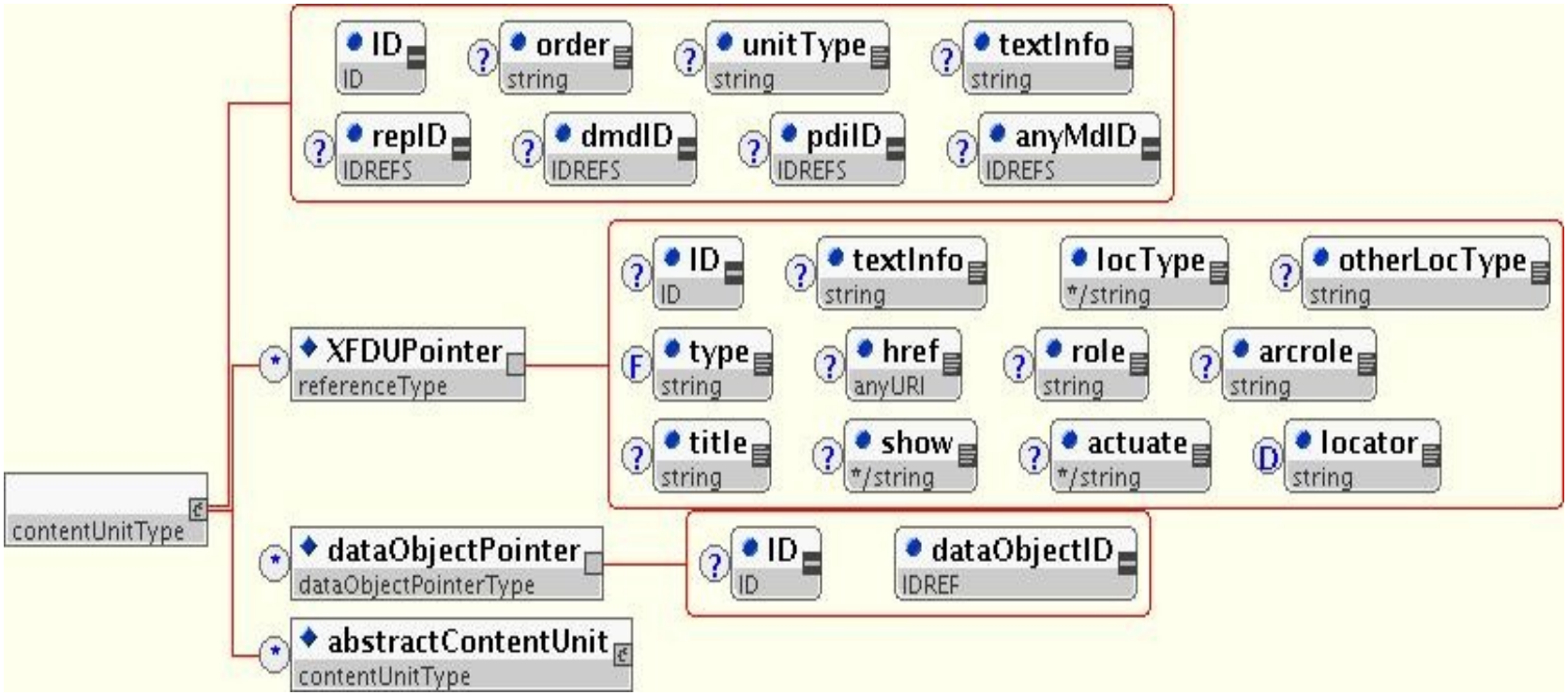
Logical View of XFDU Manifest



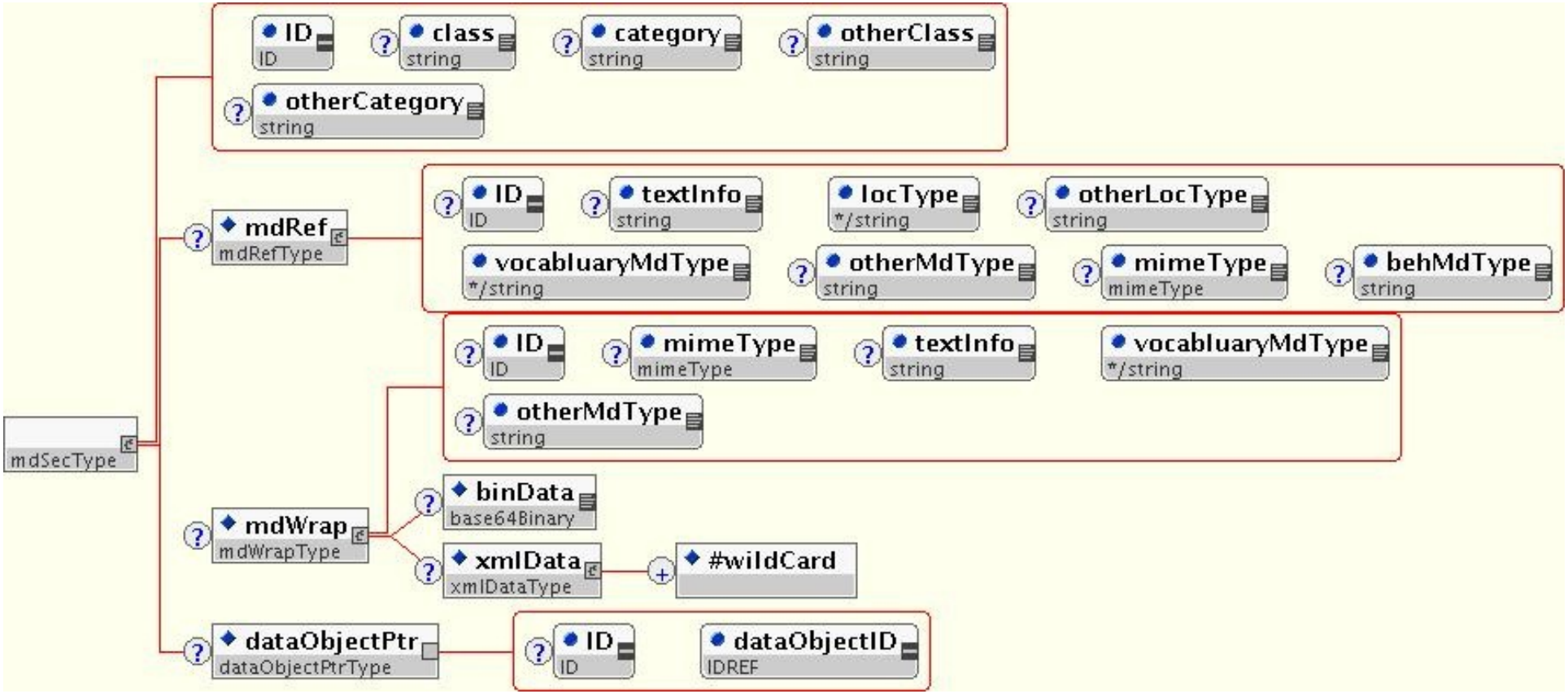
Schema Diagram of XFDU Manifest



XFDU Content Unit



XFDUMetadata Section Type



XFDU MANIFEST OVERVIEW

- **Package Header (packageHeader): Administrative metadata** for the whole XFDU Package, such as version, operating system, hardware, author, etc, and **metadata about transformations and behaviours** that must be understood, in particular, transformations which must be reversed to access the data content.
- **Metadata Section (metadataSection)**: This section records all of the metadata for all items in the XFDU package. Multiple metadata objects are allowed so that the metadata can be recorded for each separate item within the XFDU object. The **metadata schema allows the package designer to define any metadata model** by providing attributes for both metadata categories and a classification scheme for finer definition withincategories. The XFDU also provides predefined metadata categories and classes via enumerate attributes that follow the OAIS information model
- **Information Package Map (informationPackageMap)** outlines a hierarchical structure for the original object being encoded, by a series of nested **contentUnit** elements. Content units contain pointers to the data objects and to the metadata associated with those objects.
- **Data Object Section (dataObjectSection)** contains a number of dataObject elements. A Data Object contains a **bytestream** and any data required to allow the information consumer to **reverse any transformations** that have been performed on the object and restore it to the byte stream intended for the original designated community and described by the Representation metadata in the Content Unit
- **Behavior Section (behaviorSection)** may contain any number of behavior objects. Each behavior object can be used to associate executable behaviors with content in the XFDU object. A behavior object contains an **interface definition** element that represents an abstract definition of the set of behaviors

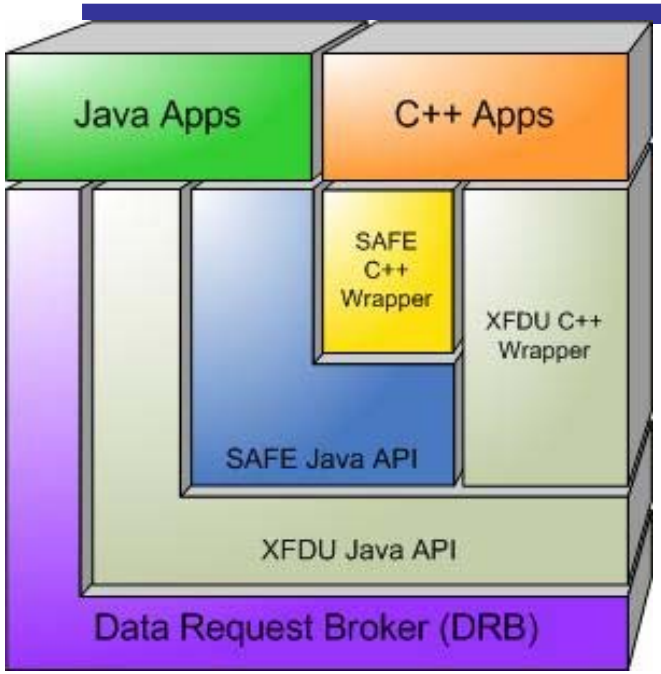
Lessons Learned in Testing

- XML schema versioning and extensibility is difficult at best
- XPath (JAVA implementation) does not scales to thousands of object

XFDU Toolkit Library

- Java class library implemented in parallel with specification development
- XFDU API library can be used within any other application that needs to perform XFDU packaging.
- To use XFDU toolkit in this manner simply put all the jar files inside of the lib directory on the CLASSPATH of your application. Javadoc documentation for the library can found at http://sindbad.gsfc.nasa.gov/xfdu
- XFDU API Library conceptually consists of two layers
  - First layer (xfdu.core.*)is low level API representing each structure in XFDU schema.
  - Second layer (xfdu.packaging.*) is more high level API. It aggregates parts of functionality from the first layer; thus, allowing easier access to constructing and manipulating an XFDU package.
  - A crude GUI is also supplied

SAFE I/O API



- **XFDU Java API**: this API provides the general features common to all XFDU packages. This API has been developed jointly with the SAFE Java API and is presently used in the framework of the activity of the CCSDS Information Packaging and Registries (IPR) Working Group in support to the validation of the XFDU recommendation developed by the group
- **XFDU C++ Wrapper**: a C++ wrapper on top of the XFDU Java API. Most of the functionality is preserved from the Java API, apart the capability of browsing the binary content
- **Data Request Broker (DRB)**: from which SAFE inherits the capability to access data independently from their formats. DRB supports in particular the SDF markup language used for annotating the XML Schema documents acting as representation information of the SAFE product objects.