Detector Description in LHCb
(Extended Version)

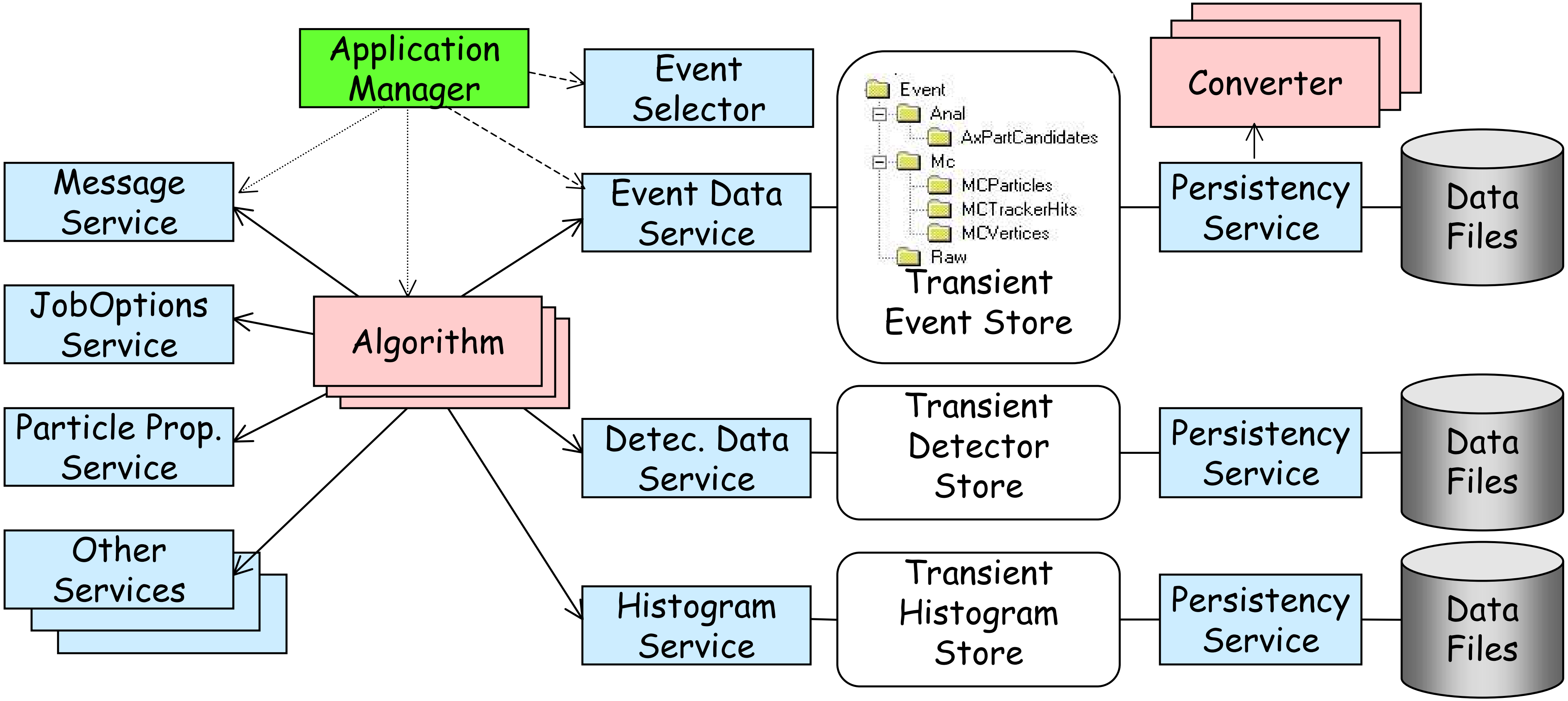
Detector Description Workshop
4 July 2002
S. Ponce - CERN

Contents

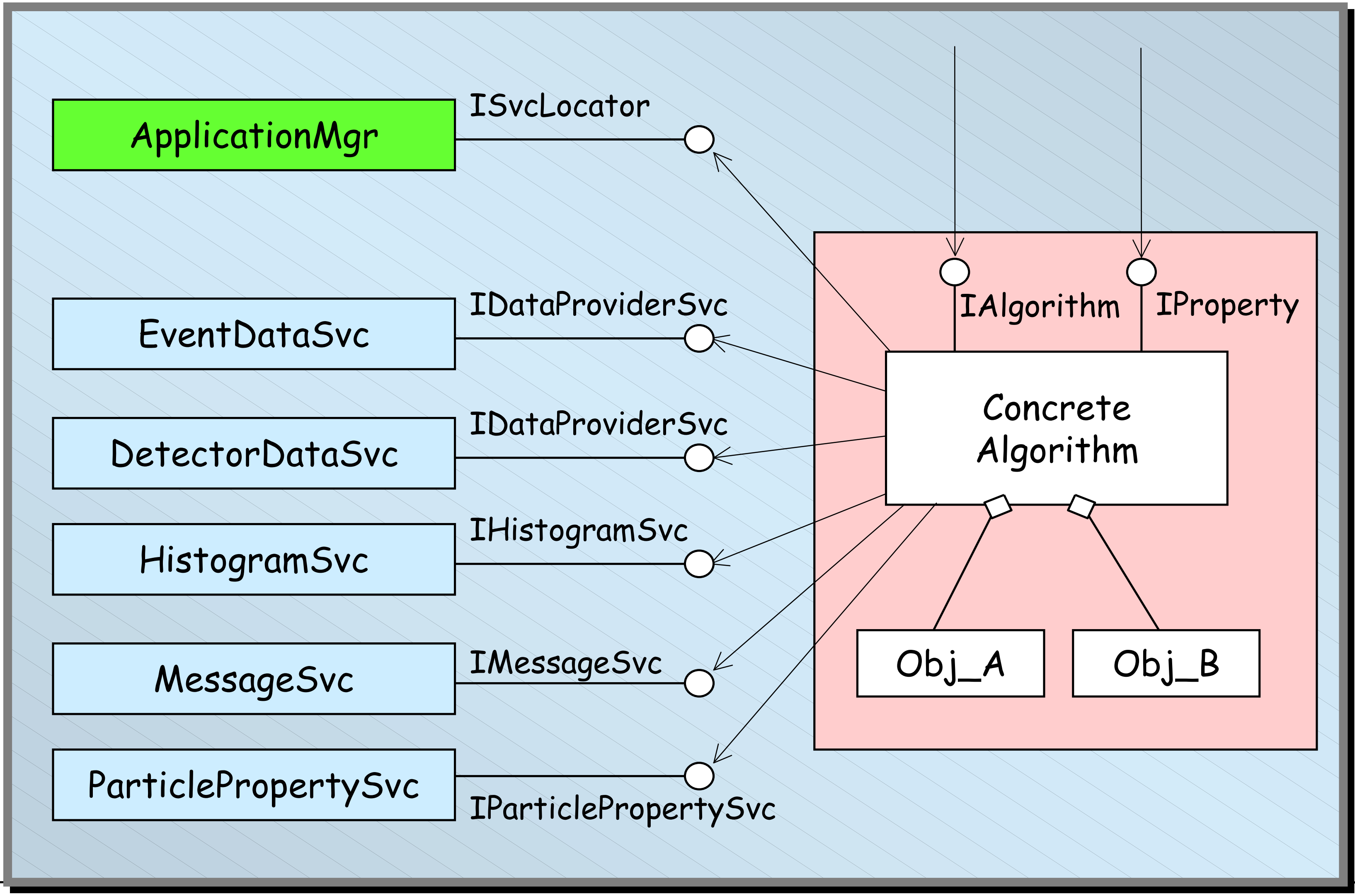
Definition of Terms

- Algorithm
  - » Atomic data processing unit (visible & controlled by the framework)
  - » Written by physicists, Called once per physics event
- Service
  - » Globally available software component providing some functionality
- Data Object
  - » Atomic data unit (visible and managed by transient data store)
- Transient Store
  - » Central service and repository for objects (load on demand)

Gaudi Object Diagram



Interfaces



ApplicationMgr
ISvcLocator
EventDataSvc
IDataProviderSvc
DetectorDataSvc
IDataProviderSvc
HistogramSvc
IHistogramSvc
MessageSvc
IMessageSvc
ParticlePropertySvc
IParticlePropertySvc
IAlgorithm
IProperty
Concrete Algorithm
Obj_A
Obj_B

Interfaces in Practice

IMyInterface.h

```
class IMyInterface {
  virtual void doSomething( int a, double b ) = 0;
}
```

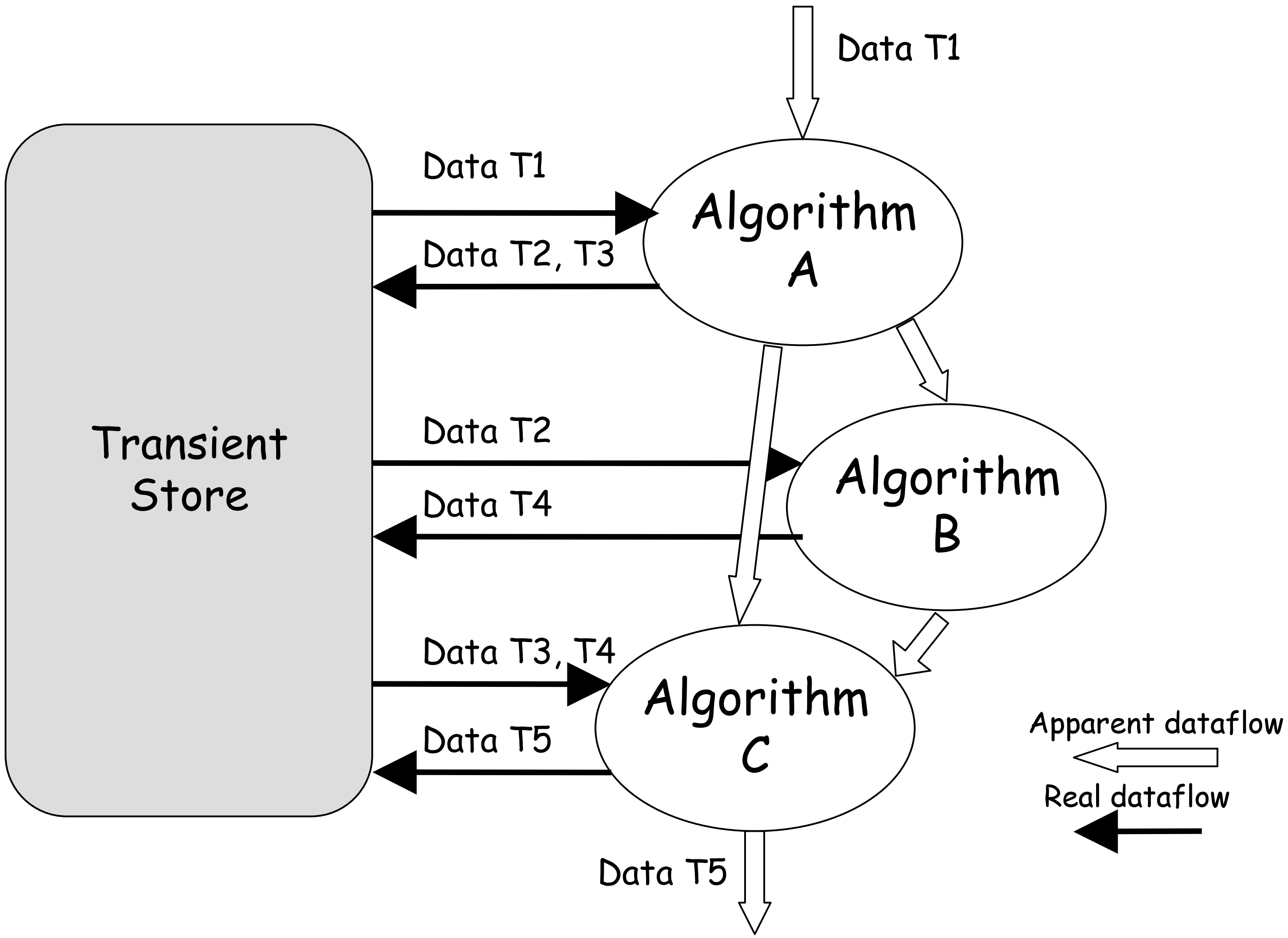
ClientAlgorithm.cpp

```
#include “IMyInterface.h”

ClientAlgorithm::myMethod() {
  // Declare the interface
  IMyInterface* myinterface;
  // Get the interface from somewhere
  service(“MyServiceProvider”, myinterface );
  // Use the interface
  myinterface->doSomething( 10, 100.5);
}
```

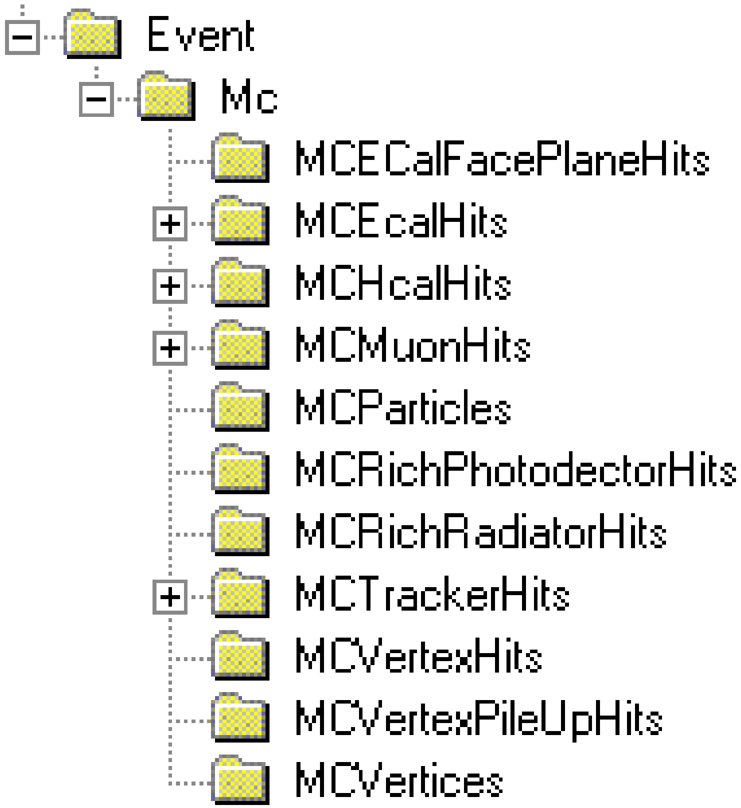
Gaudi Services

- JobOptions Service
- Message Service
- Particle Properties Service
- Event Data Service
- Histogram Service
- N-tuple Service
- Detector Data Service
- Magnetic Field Service
- Tracking Material Service
- Random Number Generator
- Chrono Service
- (Persistency Services)
- (User Interface & Visualization Services)
- (Geant4 Services)

Algorithm & Transient Store

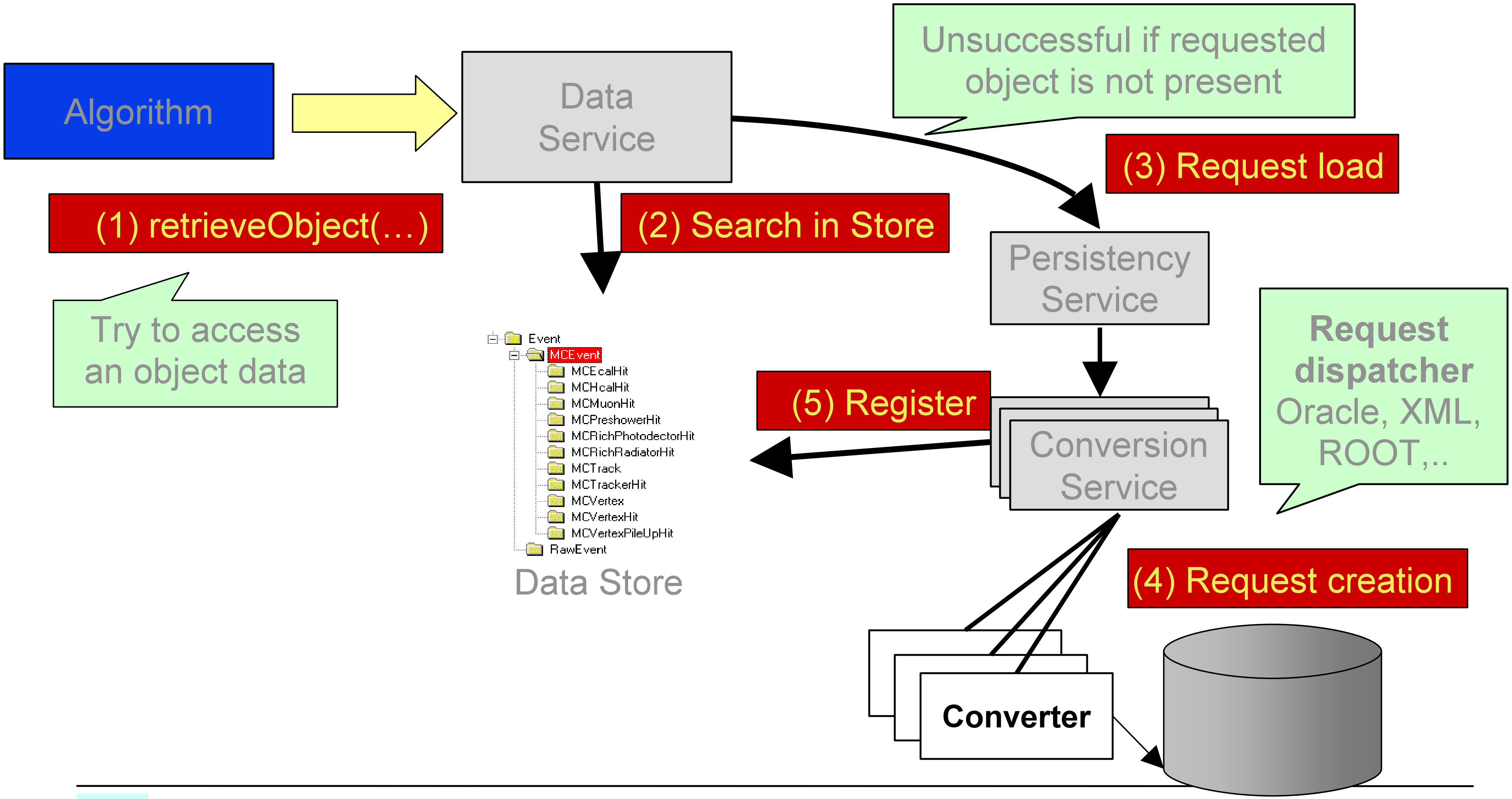


Data Reside In Data Store

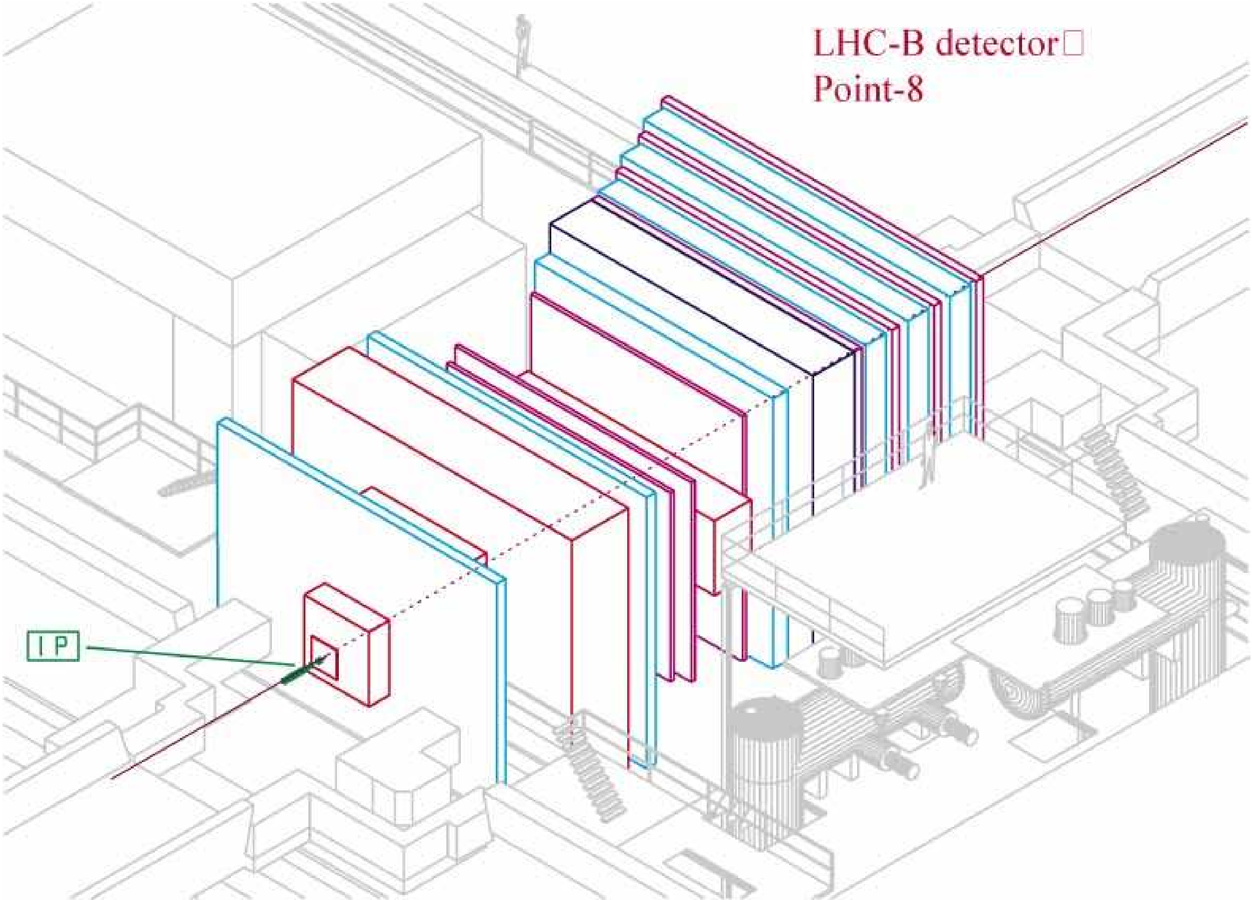


- Tree - similar to file system
- Identification by path
  "/Event/MCEvent/MCEcalHit"
  "/dd/Geometry/Ecal/Station1"
- Objects loaded on demand

Understanding Transient Store Loading



Detector Description

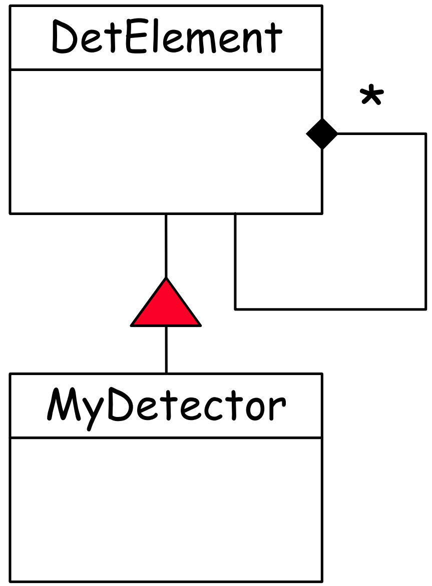


Detector Description

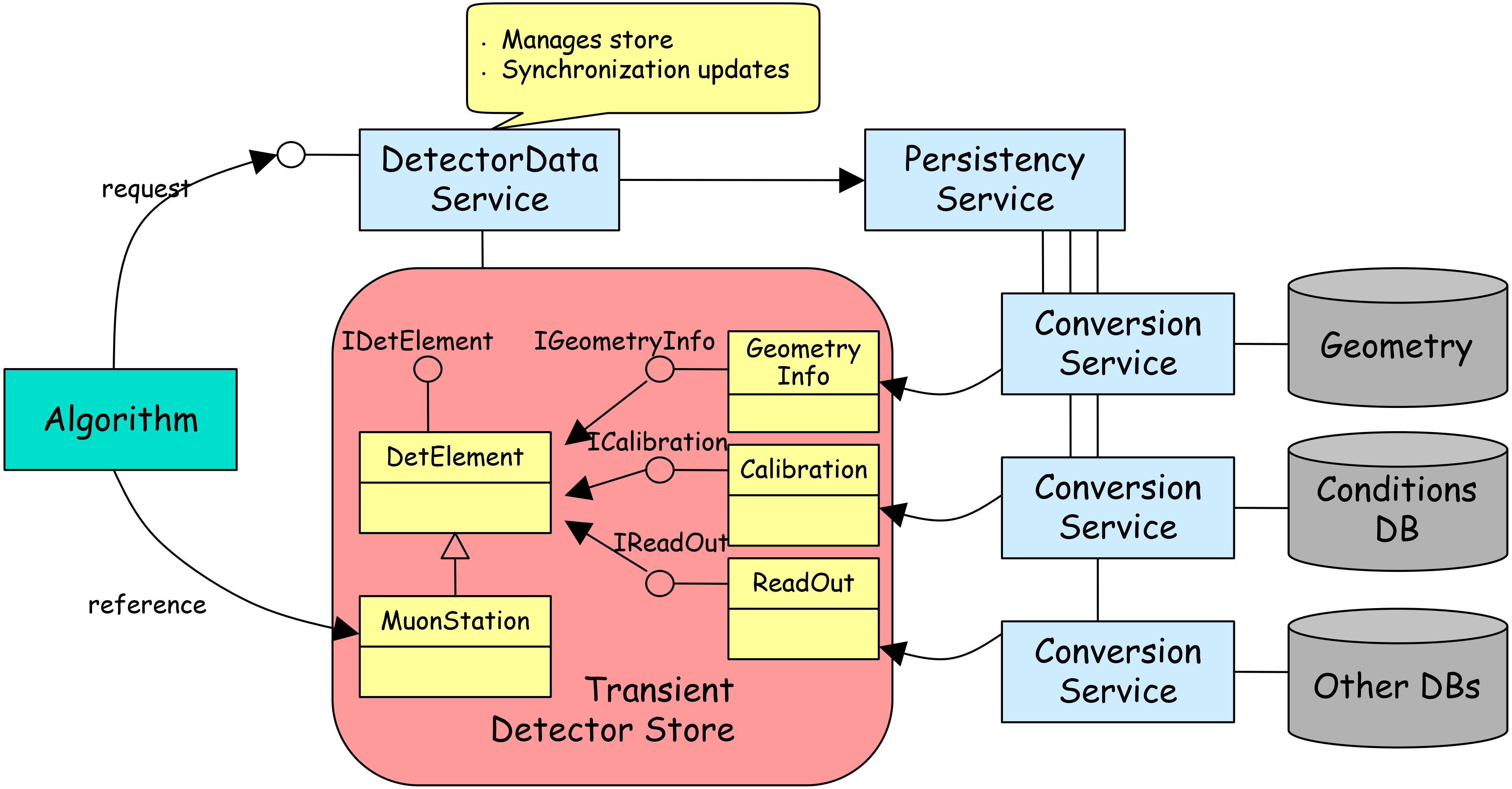
- Logical Structure
  - Breakdown of detectors
  - Identification
- Geometry Structure
  - Hierarchy of geometrical volumes
  - LogicalVolumes (unplaced dimensioned shape)
  - PhysicalVolumes (placed volume)
- Other detector data
  - Calibration, Alignment, Readout maps, Slow control, etc.

Logical Structure

- The basic object is a Detector Element
  - Identification
  - Navigation (tree-like)
- DetElement as information center
  - Be able to answer any detector related question
    - » E.g. global position of strip#, temperature of detector, absolute channel gain, etc.
  - Placeholder for specific code
    - » The specific answers will be coded by physicists



Algorithm Accessing Detector Data



Algorithm Accessing Detector Data

```
// Algorithm code fragment (initialize() or execute())

SmartDataPtr<MyDetElement> mydet(detSvc(),
                                 "Structure/LHCb/MyDet");
if( !mydet ) {
  log << MSG::ERROR << "Can't retrieve MyDet" << endmsg;
  return StatusCode::FAILURE;
}
...
// get the number of sub-DetectorElements
ndet = mydet->childIDetectorElements().size()
// get the material
material = mydet->geometry()->lvolume()->materialName();
```

Geometry Information

- Constructed using Logical and Physical Volumes (Geant 4)
  - Logical Volume: Unplaced detector described as a solid of a given material (optional) and a set of daughters (physical volumes).
  - Physical Volume: Placement of a logical volume (rotation & translation).
- Solids
  - A number of basic shapes (boxes, tubes, cones, trds, spheres,….) with dimensions
  - Boolean solids (unions, intersections and subtractions)

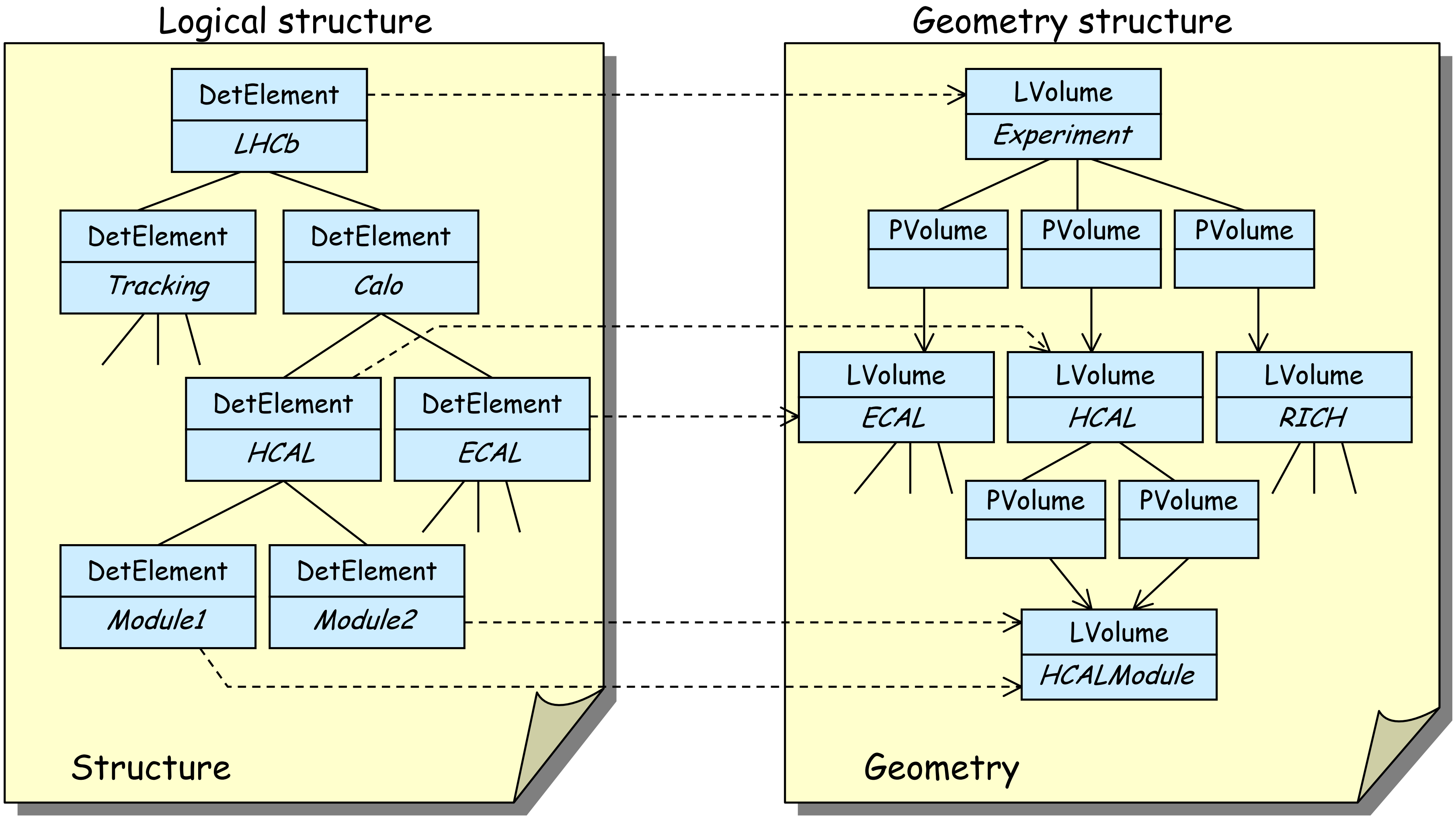
Algorithm Accessing Geometry Info

```
IGeometryInfo* geom = mydetelem->geometry();
```

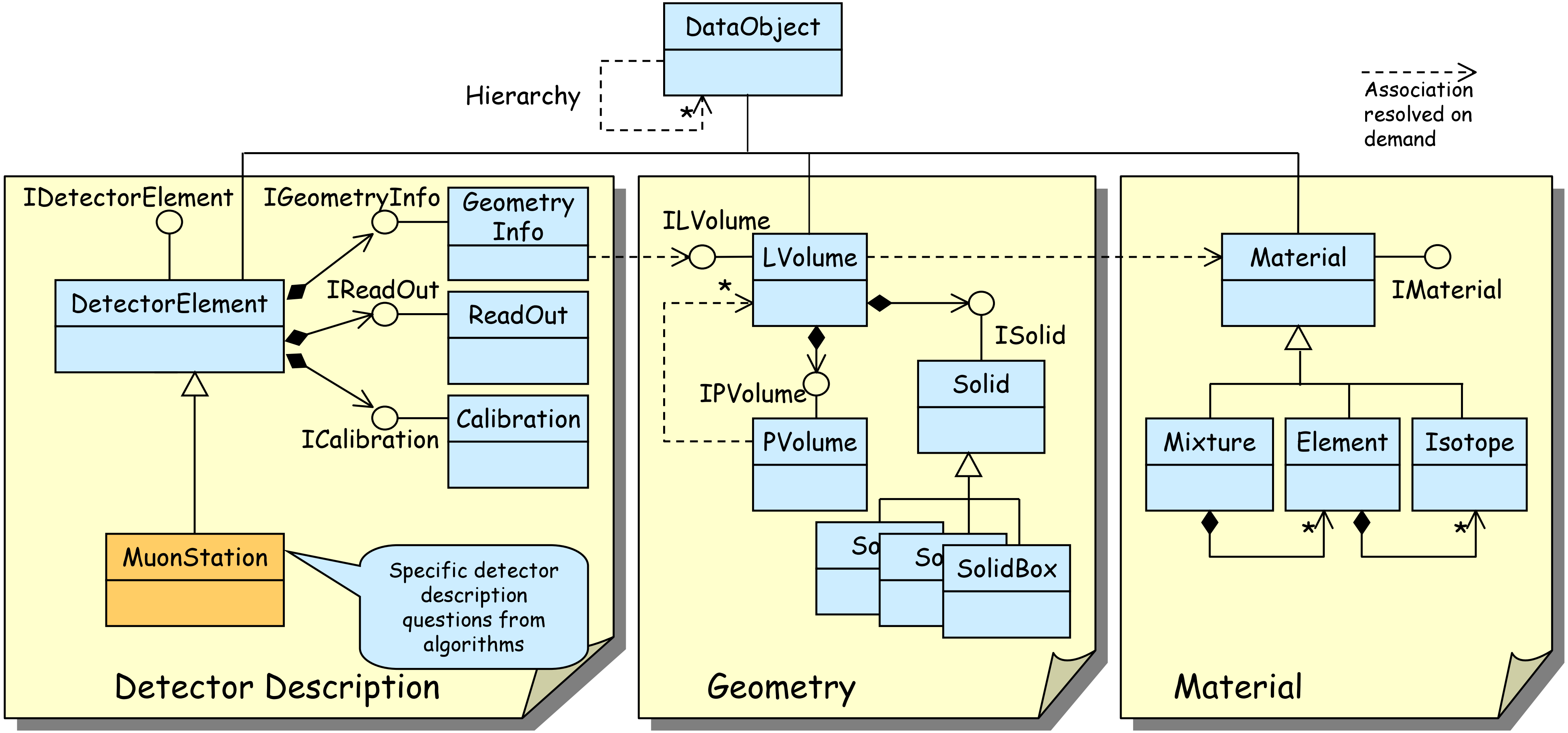
IGeometryInfo

```
HepTransform3D& matrix()        // To Local
HepTransform3D& matrixInv()     // To Global
HepPoint3D toLocal( HepPoint3D& )
HepPoint3D toGlobal( HepPoint3D& )
bool isInside( HepPoint3D& )
string belongsToPath( HepPoint3D& )
IGeometryInfo* belongsTo( HepPoint3D& )
...
fullGeoInfoForPoint( HepPoint3D&, ...)
string lVolumeName()
ILVolume* lvolume() ...
```

Two Hierarchies

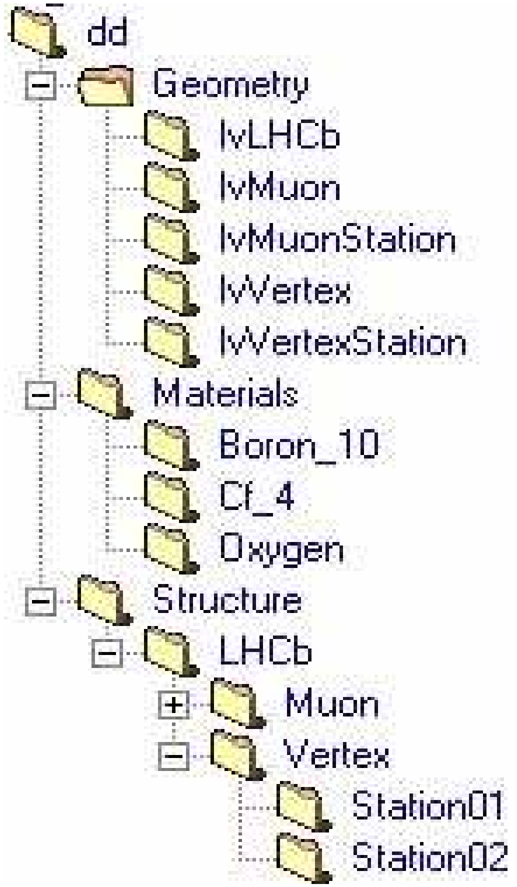


Class Diagram (Simplified)



Transient Store Organization

- Standard Gaudi Transient Store
  - "Catalogs" of Logical Volumes and Materials
  - "Structure" as a tree
  - All elements identified with names of the form: /xxx/yyy/zzzz



Persistency Based on XML Files

- XML is used as persistent representation of the Structure, Geometry and Materials
- Why XML?
  - Instead of inventing our own format use a standard one (extendible)
  - Many available Parsers and Tools
  - Strategic technology

The LHCb Detector DTD

- Divided into 3 main parts
  - » structure
  - » geometry
  - » material
- External DTDs, to be referenced in every LHCb XML files

Some Specificities

- **Expressions evaluator – units & functions known**

```
12.2*mm + .17*m / tan (34*degree)
```

- **parameter : a kind of macro**

```
<parameter name="InCell" value="40.6667*mm"/>
<parameter name="MidCell" value="1.5*InCell"/>
```

- **References : element + “ref”**

```
<detelemref href="LHCb/structure.xml#LHCb"/>
```

protocol://hostname/path/file.xml#ObjectID

Structure Elements

- DDDB : the root
- catalog : a list
- detelem : a detector element
- geometryInfo : connection to the geometry
- userParameter(Vector) : hook for adding parameters
- specific : hook for extending the DTD

```
<DDDB>
 <catalog name="…">
  <detelem name="…">
   <geometryinfo
       lvname="…"
       npath="…"
       support="…"/>
   <userParameter
       comment="…"
       name="…"
       type="string">
    …
   </userParameter>
   <specific>
    …
   </specific>
  </detelem>
 </catalog>
</DDDB>
```

Geometry Elements (1)

- DDDB : **the root**
- catalog : **a list**
- logvol : **logical volume**
- physvol : **physical volume**
- paramphysvol(2D)(3D) : **replication of physical volumes**
- tabproperty : **tabulated properties**

```
<DDDB>
 <catalog name="…">
  <logvol material="…"
          name="…">
   <physvol logvol="…"
            name="…"/>
  </logvol>
  <logvol name="…">
   <paramphysvol number="5">
    <physvol logvol="…"
             name="…"/>
    <posXYZ z="20*cm"/>
   </paramphysvol>
  </logvol>
 </catalog>
</DDDB>
```

Geometry Elements(2)

- posXYZ, posRPhiZ, posRThPhi : **translations**
- rotXYZ, rotAxis : **rotations**
- transformation : **composition of transformations**
- box, trd, trap, cons, tub, sphere, polycon
- union, intersection, subtraction : **boolean solids**
- surface

```
<subtraction name="sub2">
 <box name="box3"
      sizeX="1*m"
      sizeY="1*m"
      sizeZ="15*cm"/>
 <tubs name="tub2"
       outerRadius="15*cm"
       sizeZ="25*cm"/>
</subtraction>
<posXYZ z="-40*cm"/>
<rotXYZ rotX="90*degree"/>
```

Material Elements

- materials **: the root**
- catalog **: a list**
- tabproperty **: tabulated properties**
- atom
- isotope
- element **: a mixture of isotopes**
- material **: mixtures of elements or materials**

```
<isotope A="11*g/mole"
         name="Bor_11" …/>
<element name="Boron"
         symbol="B" …>
  <isotoperef href="#Bor_10"
        fractionmass="0.20"/>
  <isotoperef href="#Bor_11"
        fractionmass="0.80"/>
</element>
<element name="Oxygen"
         symbol="O" …>
  <atom A="16*g/mole"
        Zeff="8.0000"/>
</element>
<material name="CO2" …>
  <component name="Carbon"
             natoms="1"/>
  <component name="Oxygen"
             natoms="2"/>
</material>
```

XmlEditor

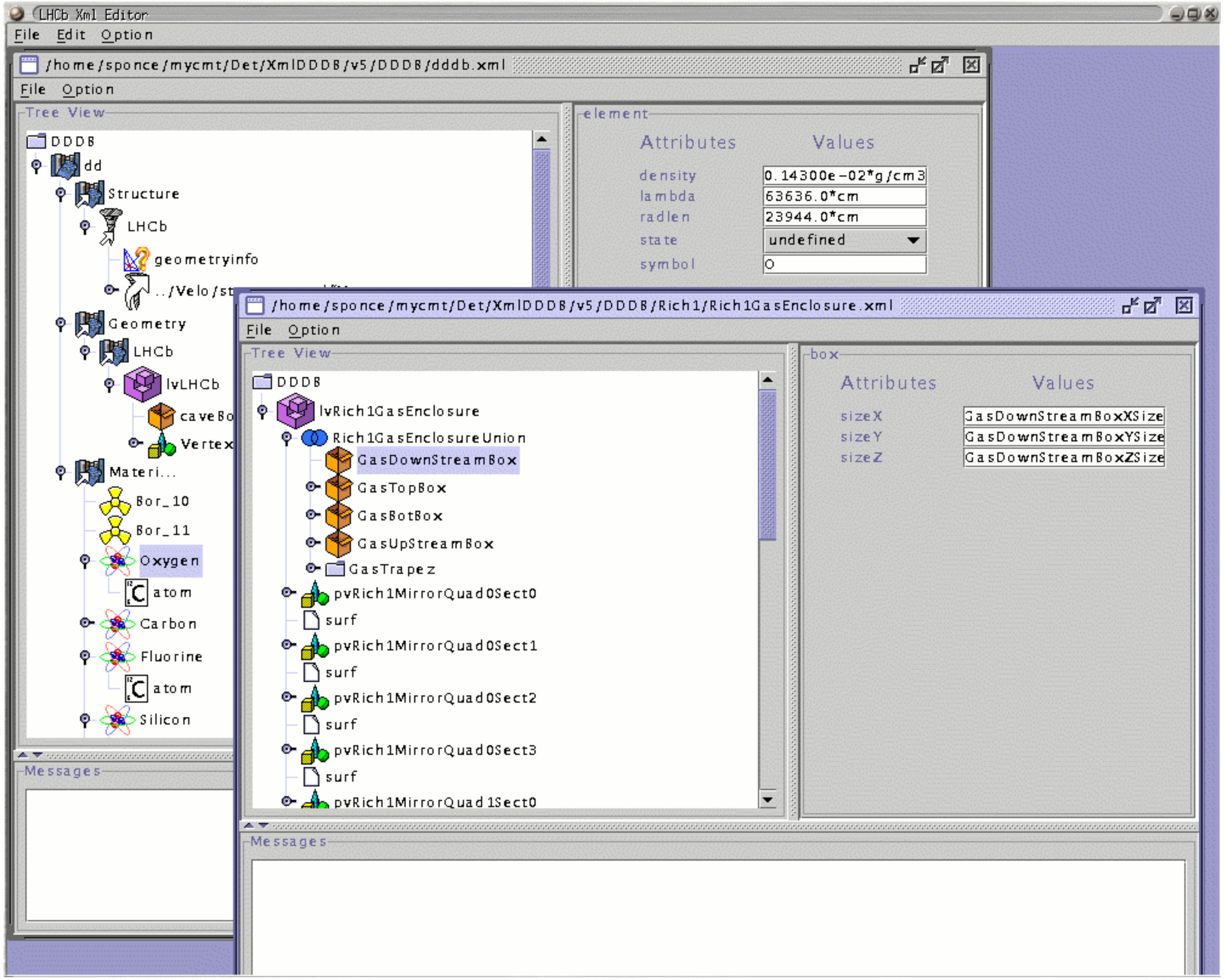
- Explorer-like XML viewer
- No need to know XML syntax
- Checks the DTD when opening a file
- Allows copy, paste and drag and drop of nodes
- Allows view of several files at the same time
- Hide references across files

→ **Easy XML edition**

$LHCBSOFT/Det/XmlEditor/v*/scripts/xmlEditor(.bat)

http://lhcb-comp.web.cern.ch/lhcb-comp/Frameworks/DetDesc/Documents/XmlEditor.pdf

XMLEditor


LHCb Xml Editor
File Edit Option
/home/sponce/mycmt/Det/XmlDDDB/v5/DDDB/dddb.xml
File Option
Tree View
DDDB
dd
Structure
LHCb
geometryinfo
../Velo/st
Geometry
LHCb
lvLHCb
caveBo
Vertex
Materi...
Bor_10
Bor_11
Oxygen
atom
Carbon
Fluorine
atom
Silicon
Messages
element
Attributes
Values
density
0.14300e-02*g/cm3
lambda
63636.0*cm
radlen
23944.0*cm
state
undefined
symbol
O
/home/sponce/mycmt/Det/XmlDDDB/v5/DDDB/Rich1/Rich1GasEnclosure.xml
File Option
Tree View
DDDB
lvRich1GasEnclosure
Rich1GasEnclosureUnion
GasDownStreamBox
GasTopBox
GasBotBox
GasUpStreamBox
GasTrapez
pvRich1MirrorQuad0Sect0
surf
pvRich1MirrorQuad0Sect1
surf
pvRich1MirrorQuad0Sect2
surf
pvRich1MirrorQuad0Sect3
surf
pvRich1MirrorQuad1Sect0
box
Attributes
Values
sizeX
GasDownStreamBoxXSize
sizeY
GasDownStreamBoxYSize
sizeZ
GasDownStreamBoxZSize
Messages

Conversion From XML to C^{++}

- Converters used to build C^{++} objets from XML
- One converter per object type
  - **XmlDetectorElementCnv**
  - **XmlLVolumeCnv**
  - **XmlMixtureCnv**
  - **XmlMuonStationCnv**
  - …
  - **XmlMySubDetCnv**
- almost 1 to 1 mapping between XML elements and C^{++} objects
- Uses the xerces-C parser - Could use any DOM parser

First Summary

- We are able to reach the geometry description from the C++ transient world
- Everything is transparent for the C++ user, there is no need to know it comes from XML

- At this point, we have no way to extend the schema and especially to add specific parameters to a detector element

Specializing Detector Elements

1. adding userParameter(vector)s to default DetectorElements
2. extending and specializing the DetectorElement object in C++, using userParameters in XML
3. extending XML DTD and writing a dedicated converter

Specializing by using UserParameter[Vector]

- Two elements :
  <userParameter> and <userParameterVector>
- 3 string attributes : name, type and comment
- One value given as text

```
<userParameter
        comment="blablabla"
        name="description"
        type="string">
  Calibration channels
</userParameter>
```

```
<userParameterVector
        name="NbChannels"
        type="int"
        comment="blabla">
   530  230
   570 270
</userParameterVector>
```

C++ API for userParameters

- Methods on DetectorElement for userParameters :
  - `string userParameterAsString (string name)`
  - `double userParameterAsDouble (string name)`
  - `int userParameterAsInt (string name)`
- The equivalent exist for userParameterVectors

```
std::string description = elem->userParameterAsString ("description");
std::vector<int> channelNbs = elem->userParameterVectorAsInt ("NbChannels");

log << MSG::INFO << description << " : “;
for (std::vector<int>::iterator it = channelNbs.begin();
      it != channelNb.end();
      it++)
  log << *it;
log << endreq;
```

Extending Detector Elements

- Free extension of the DetectorElement class
- Specific initialization using initialize()
  - called after conversion
  - access to userParameters
- A converter is needed but very simple (4 lines)

```
#include "DetDesc/XmlUserDetElemCnv.h"
#include "MyDetElem.h"

static CnvFactory
  <XmlUserDetElemCnv<MyDetElem> > s_factory;
const ICnvFactory& XmlMyDetElemCnvFactory = s_factory;
```

Full Customization

- extension of the DTD to define new XML elements
- parsing of the new XML code using the xerces parser
- “real” converters to initialize C^{++} objects according to XML

The <Specific> Element

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DDDB SYSTEM “extendedDtd.dtd”>
<DDDB>
  <detelem classID="7294" name="Head">
    <geometryinfo …/>
    <specific>
      <channelSet description=“…" name="Controls">
        <channels description="Inputs" nb="20"/>
        <channels description="Outputs" nb="150"/>
      </channelSet>
      <channelSet description=“…" name="Data">
        <channels description="head" nb="2000"/>
      </channelSet>
    </specific>
  </detelem>
</DDDB>
```

Writing a Converter

- One needs :
  - to get a C^{++} representation of the XML (DOM tree)
  - to deal with expressions and parameters
  - to reuse existing code (only convert specific XML elements !!!)

Implementing the Converter

- Real converter =
1. extension of XmlUserDetElemCnv<DeType>
2. implementation of method

   StatusCode i_fillSpecificObj (DOM_Element, DeType*)

   - **`i_fillSpecificObj`** is called once per direct child of tag <specific>
   - the DOM_Element is given, the DeType object was created and must be populated
   - all other elements (not inside <specific>) are automatically converted

Converter Example (1)

```
class XmlMyDetElemCnv :
  public XmlUserDetElemCnv<MyDetElem> {

public:
  XmlMyDetElemCnv (ISvcLocator* svc);
  ~XmlMyDetElemCnv() {}

protected:
  virtual StatusCode i_fillSpecificObj
    (DOM_Element childElement,
     MyDetElem* dataObj);
};

static CnvFactory<XmlMyDetElemCnv> s_Factory;
const ICnvFactory& XmlMyDetElemCnvFactory = s_Factory;

XmlMyDetElemCnv::XmlMyDetElemCnv(ISvcLocator* svc) :
  XmlUserDetElemCnv<MyDetElem> (svc) {}
```

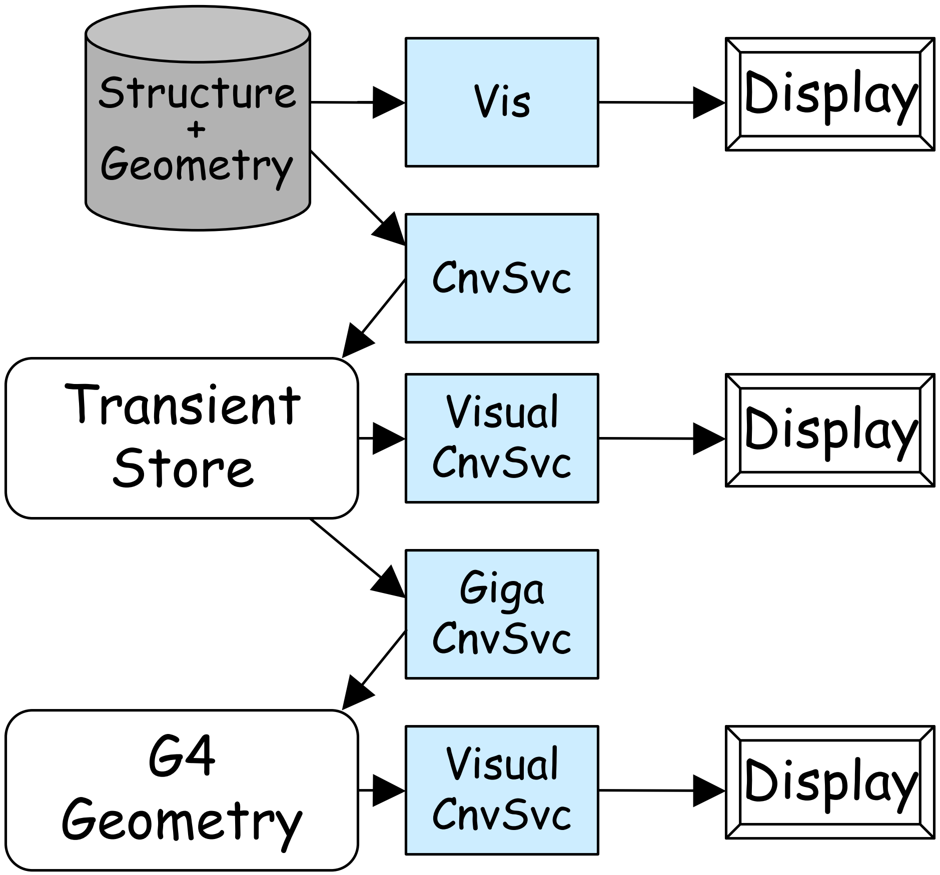
Converter Example (2)

```
StatusCode XmlMyDetElemCnv::i_fillSpecificObj
  (DOM_Element childElement, MyDetElem* dataObj) {

  std::string elementName =
     dom2Std (childElement.getNodeName());

  if ("channelSet" == elementName) {
    const std::string name = dom2Std
      (childElement.getAttribute ("name"));
    const std::string description = dom2Std
      (childElement.getAttribute ("description"));
    dataObj->addChannelSet(name, description);
    …
  } else {
    …
}
```

Panoramix

Geometry Visualization

- Visualization is essential for developing the geometry
  - Applicable at the different data representations
- Generic geometry information conversion to 3D graphics data
- Panoramix (OnX)



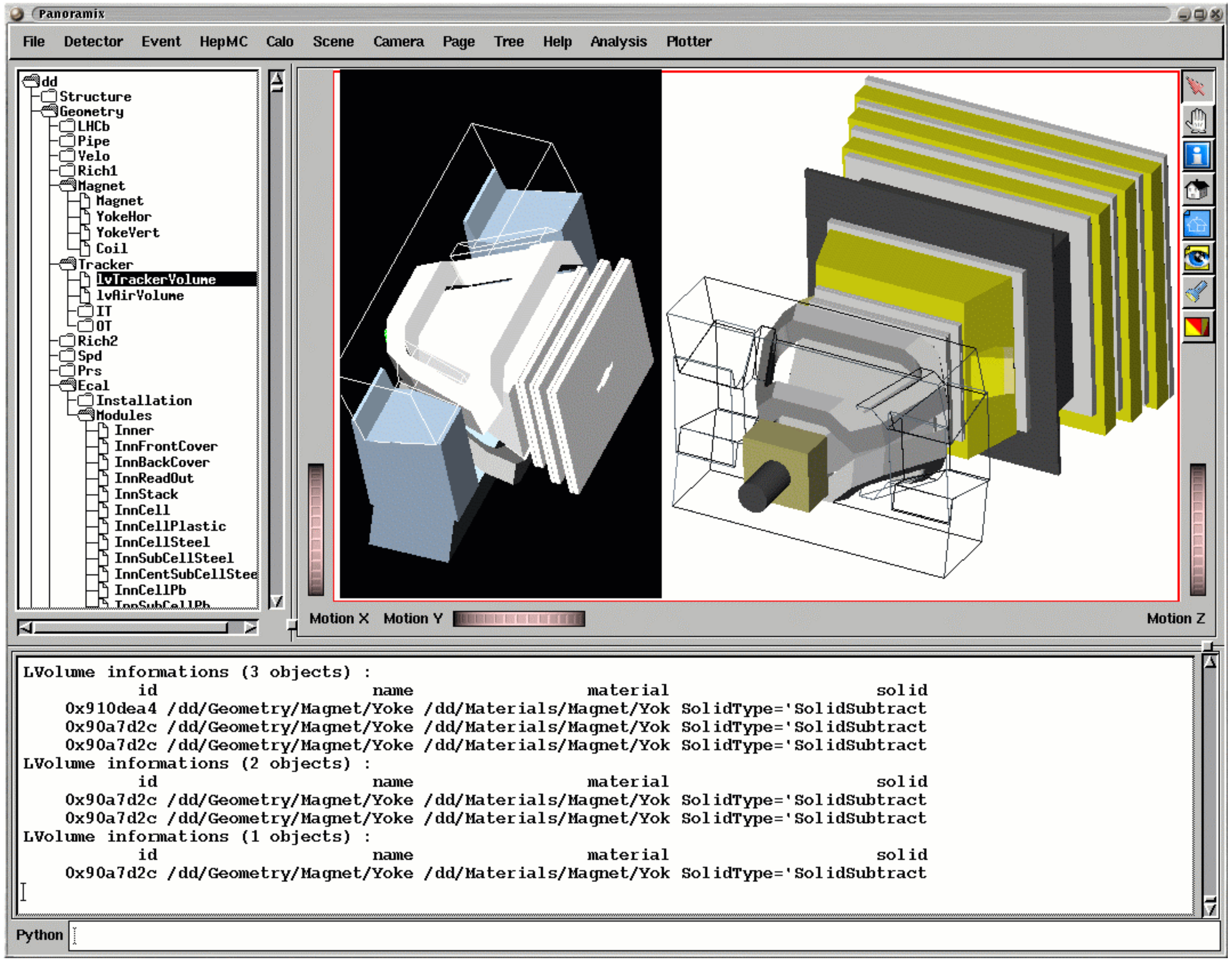
Panoramix

- Events and Geometry viewer
- Takes LHCb specificities into account
  - » references
  - » logical volumes hierarchy
  - » subDetectors
- Interactive move inside the geometry

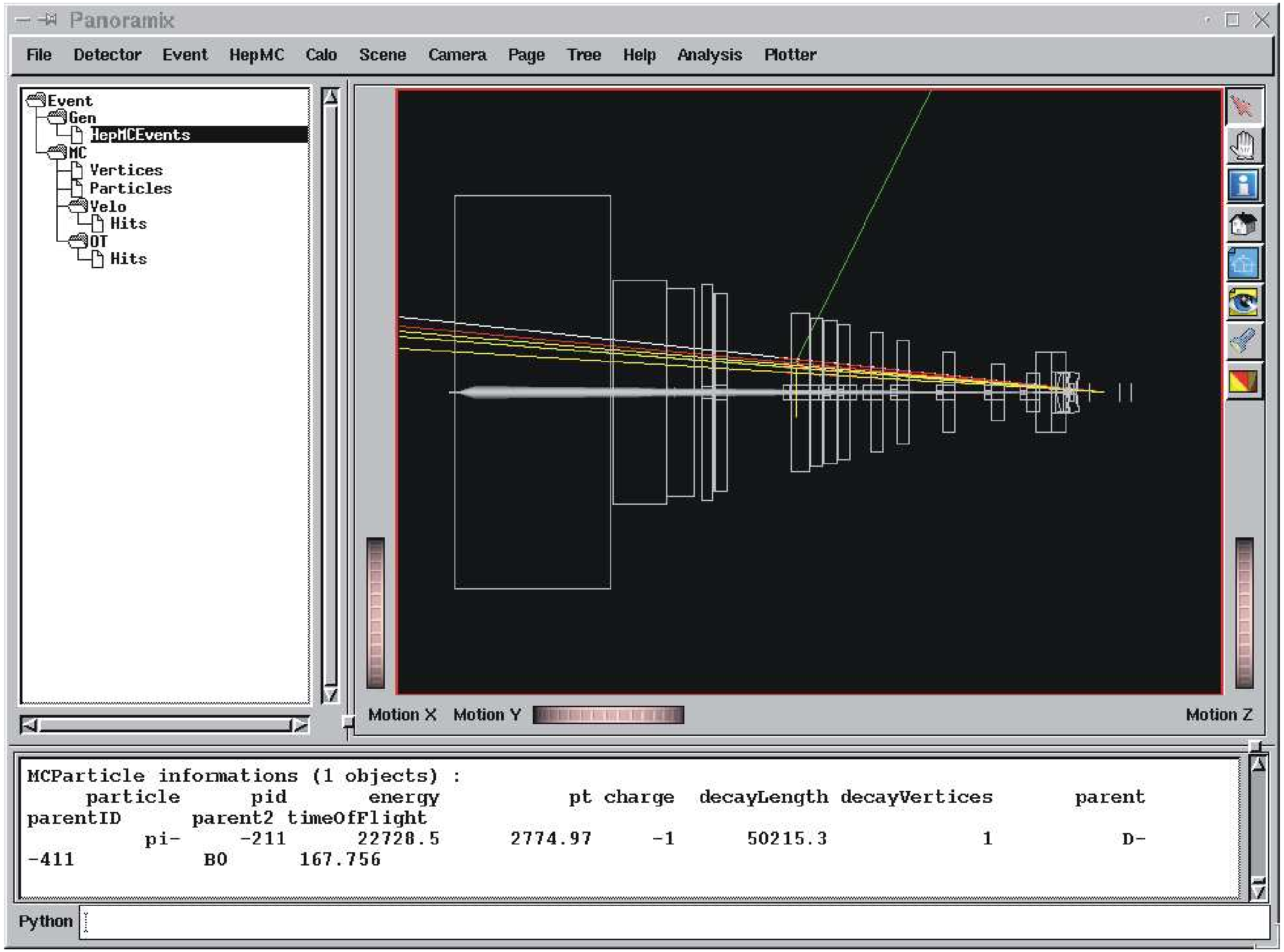
$LHCBSOFT/Vis/Panoramix/v*/scripts/panoramix(.bat)

http://www.lal.in2p3.fr/SI/Panoramix/tutorial/tutorial.html

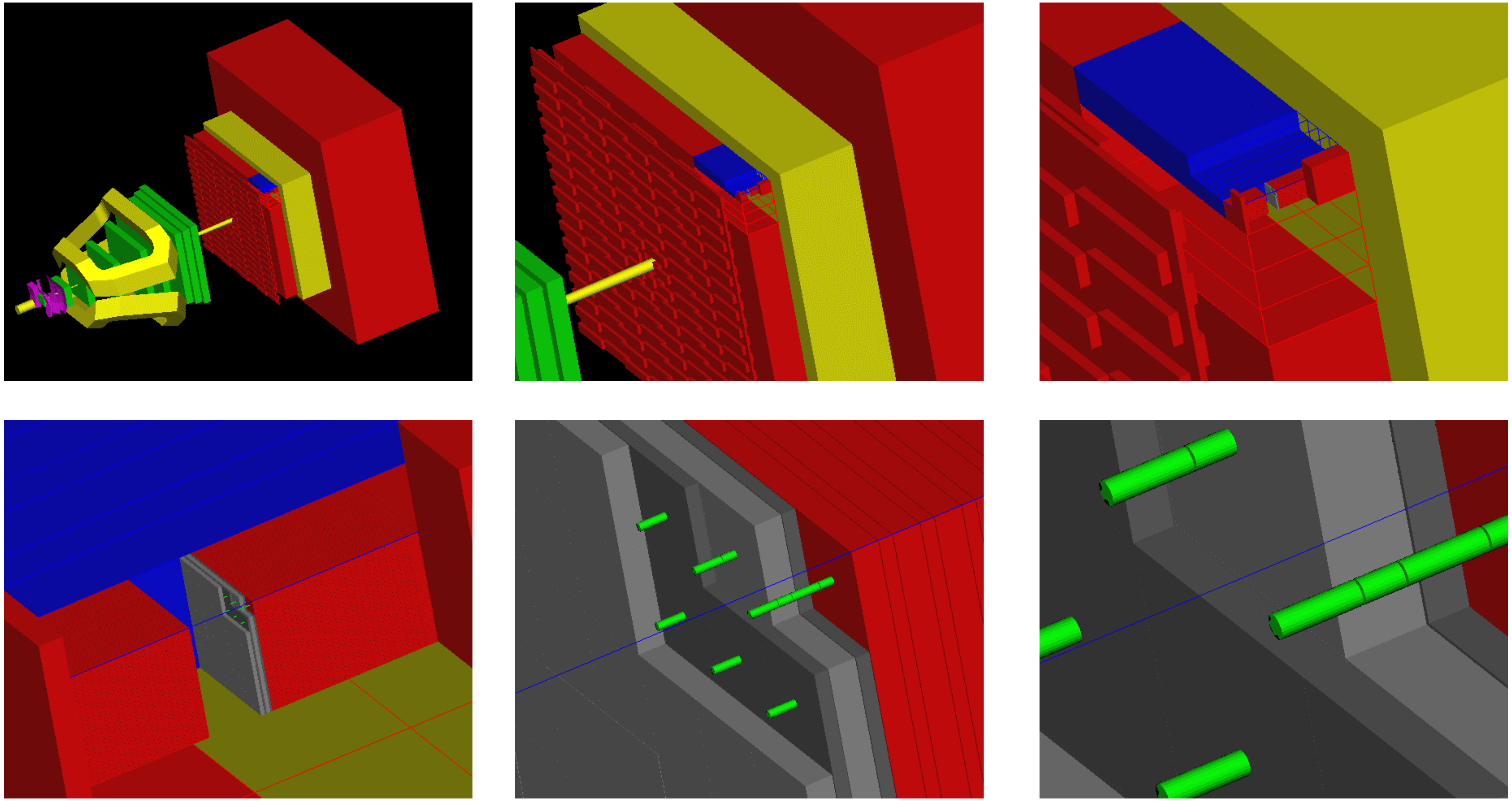
Panoramix GUI



Event Visualization



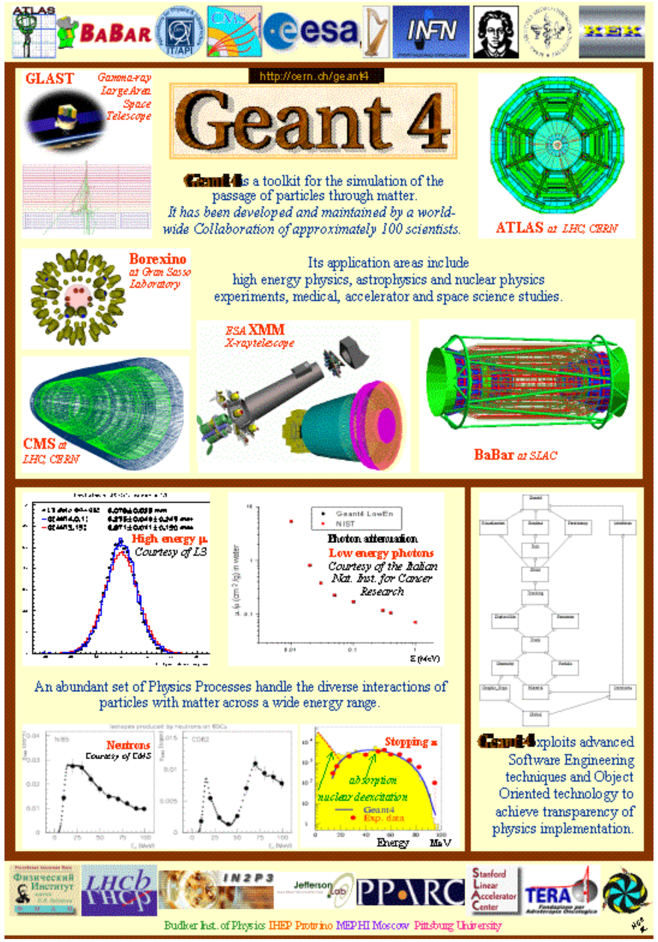
Zoom on Ecal



The VisualizationSvc

- A Gaudi service
- Used by Panoramix/Geant4 converters
- Allows independent customisation of visualization, shared by all visualization softwares
- Takes into account :
  - colors (with alpha channel)
  - visibility
  - open status
  - display mode (wire Frame, Plain)

Geant4



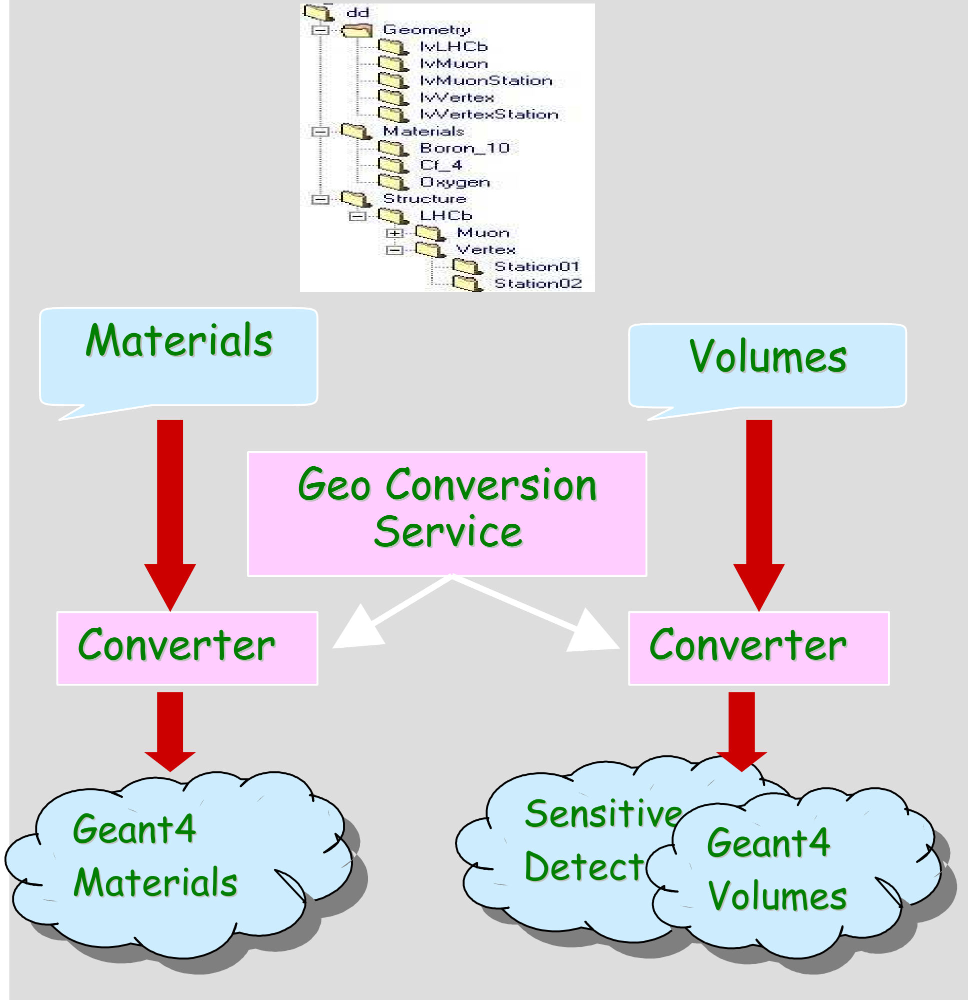
http://cern.ch/geant4
Geant 4
Geant4 is a toolkit for the simulation of the passage of particles through matter.
It has been developed and maintained by a world-wide Collaboration of approximately 100 scientists.
GLAST Gamma-ray Large Area Space Telescope
ATLAS at LHC CERN
Borexino at Gran Sasso Laboratory
Its application areas include high energy physics, astrophysics and nuclear physics experiments, medical, accelerator and space science studies.
ESA XMM X-ray telescope
CMS at LHC CERN
BaBar at SLAC
High energy μ
Courtesy of L3
Photon attenuation
Low energy photons
Courtesy of the Italian Nat. Inst. for Cancer Research
An abundant set of Physics Processes handle the diverse interactions of particles with matter across a wide energy range.
Neutrons
Courtesy of CMS
Stopping π
absorption
nuclear deexcitation
Geant4
Exp. data
Energy MeV
Geant4 exploits advanced Software Engineering techniques and Object Oriented technology to achieve transparency of physics implementation.
Budker Inst. of Physics IHEP Protvino MEPHI Moscow Pittsburg University

Interfacing With Geant4

- We integrate Gaudi with Geant4 by providing a number of “Gaudi Services” (GiGa)
- The GiGaGeomCnvSvc is able to convert transient objects (DetElem, LVolume, Surfaces, etc.) into G4 geometry objects
  - The conversion does not require “user” code
  - Flexibility in mapping Gaudi model to Geant4 model
- Single source of Geometry information

GiGa Geometry Conversion

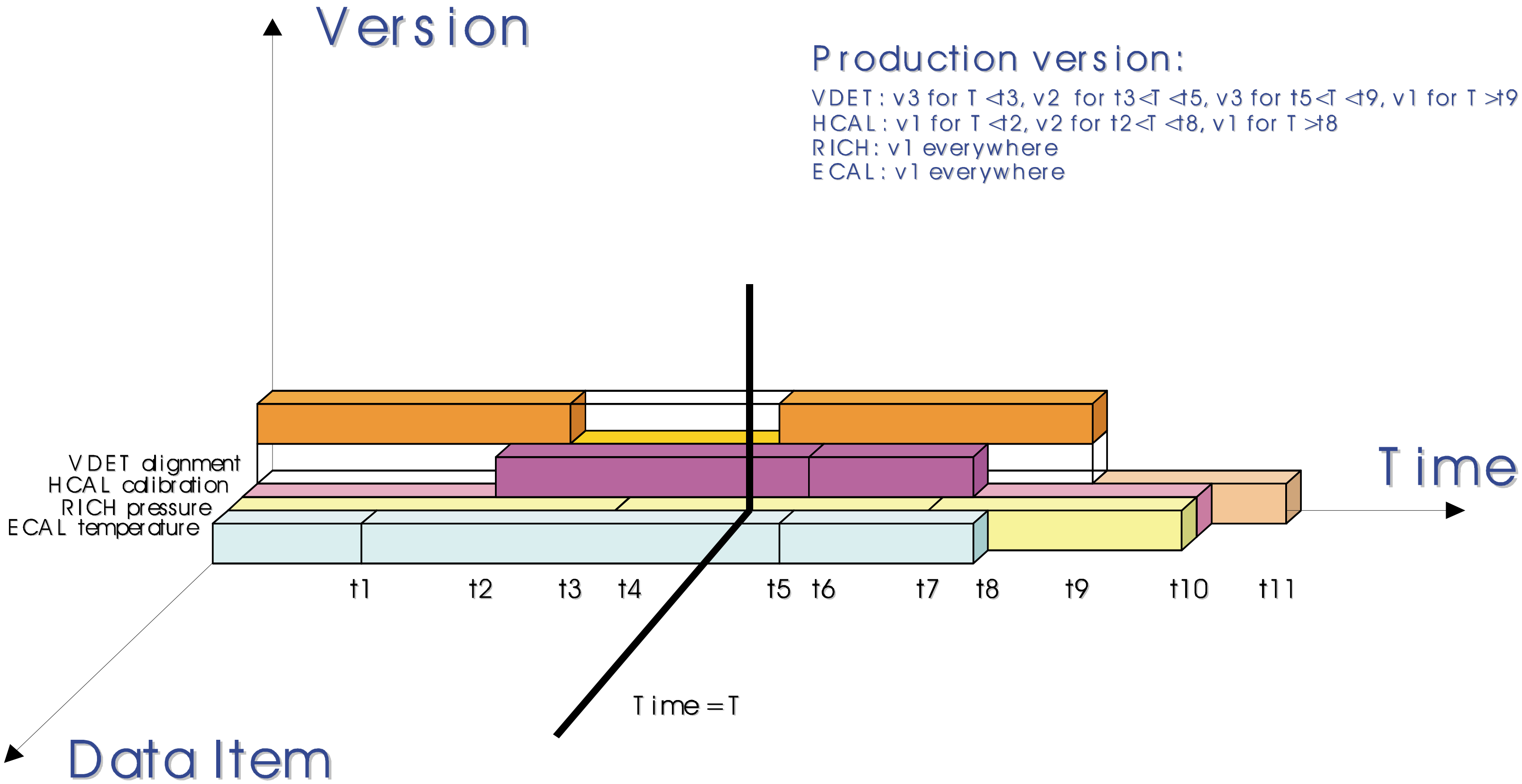
- Unidirectional
- Conversion of transient detector description (common) into Geant4 representation
- Gaudi Conversion Service and Converters
  - Volumes & Surfaces
  - Materials
- Instantiation of Sensitive Detector and Magnetic Field objects through Abstract Factory pattern



Condition Database

Production version:

VDET: v3 for T < t3, v2 for t3 < T < t5, v3 for t5 < T < t9, v1 for T > t9
HCAL: v1 for T < t2, v2 for t2 < T < t8, v1 for T > t8
RICH: v1 everywhere
ECAL: v1 everywhere



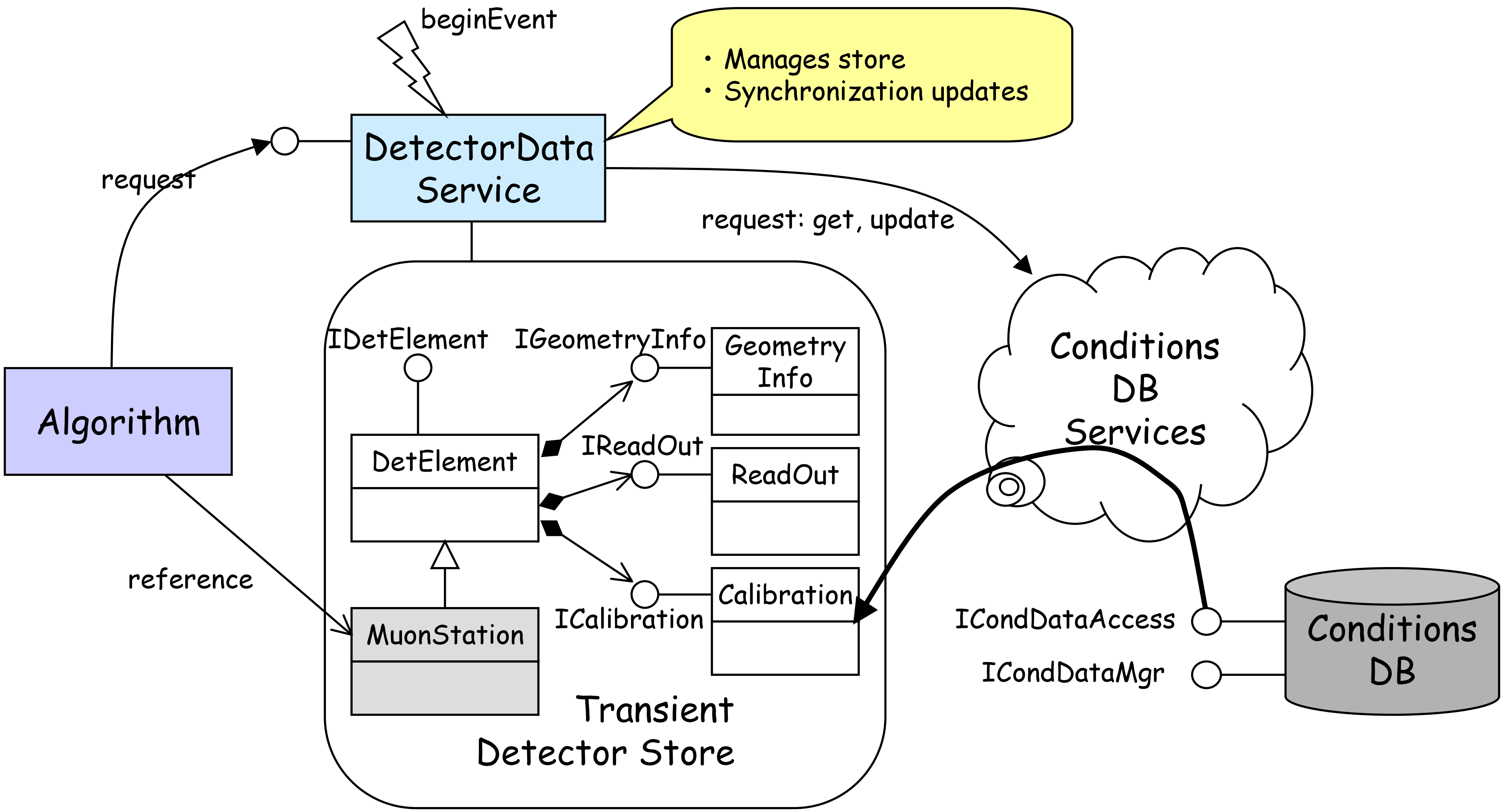
Conditions DB

- Detector conditions data (calibration, slow control, alignment, etc.) are characterized by:
  - » Time validity period
  - » Version
- The conditions data objects will also appear in the Detector Transient Store
- The persistency of conditions data is done with the Conditions DB (IT product)

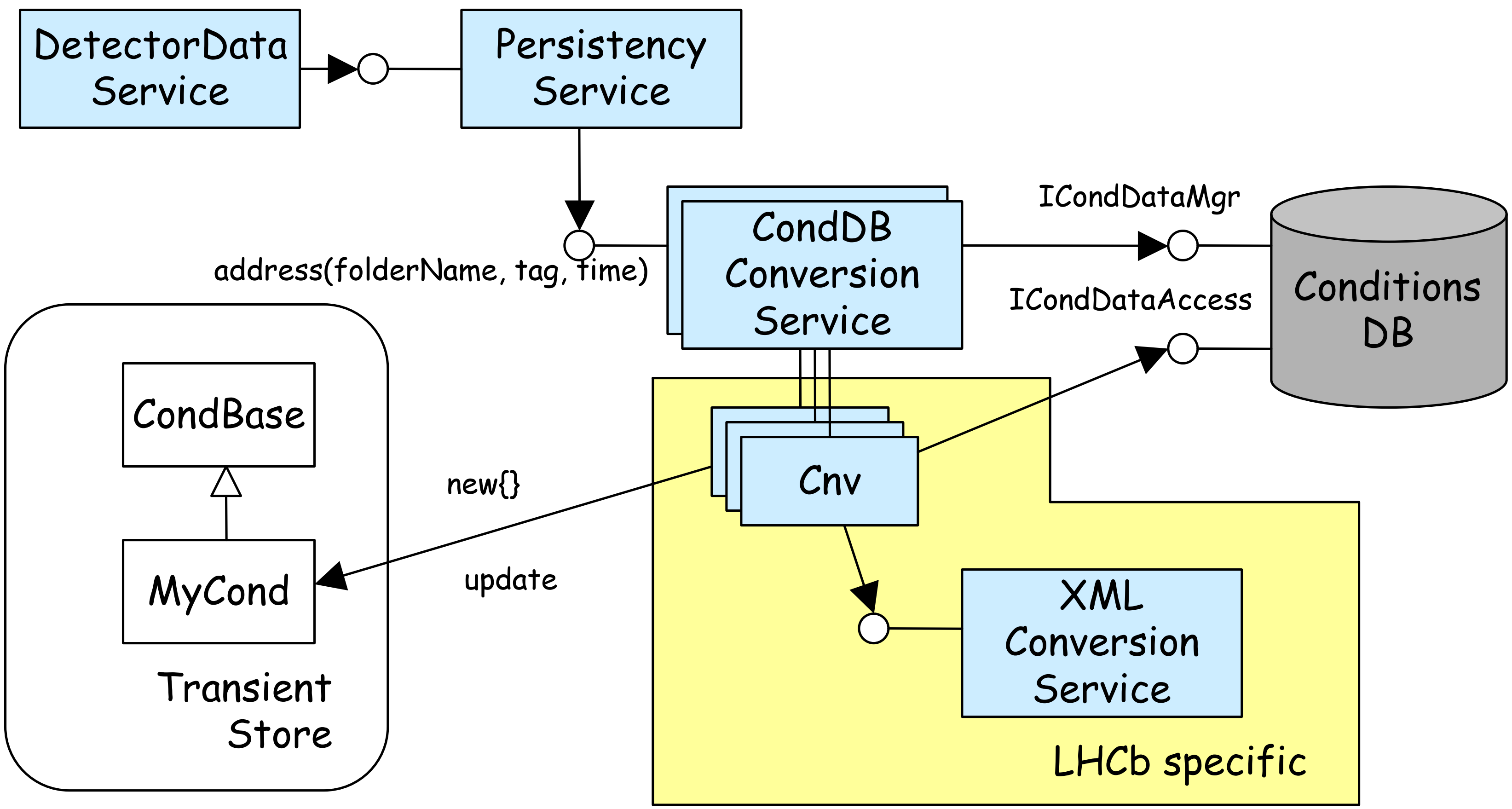
Condition Data Object

- "Block" of data belonging to some detector element
  - coded in XML
  - seen as a BLOB by the database
- Time (CondDBKey) validity range
  - [since, till]
  - CondDBKey is a 64 bit integer number. Sufficient flexibility (absolute time in ns, run number, etc.)
- Version
  - Sequence version number
- Extra information
  - Textual description, insertion time, etc.

ConditionsDB: Integration in Gaudi



Conditions Conversion Service



Conditions DB Implementation

- The databse used is ORACLE through the IT implementation of the interface already used for objectivity.
- XML references are used to select between plain XML and condition DB :
  - <conditionref href="../Ecal/condition.xml#caEcal"/> → XML
  - <conditionref href="cond://dd/Calibration/Ecal/caEcal"/> → DataBase