Laboratoire de l'Accélérateur Linéaire (IN2P3-CNRS) Orsay, France

Olivier Callot

6 April 2000

The Calorimeter Detector Description

- The calorimeter problem
- Detector element and volumes
- Objects to produce
- Suggested XML and C++ improvements

Disclaimer

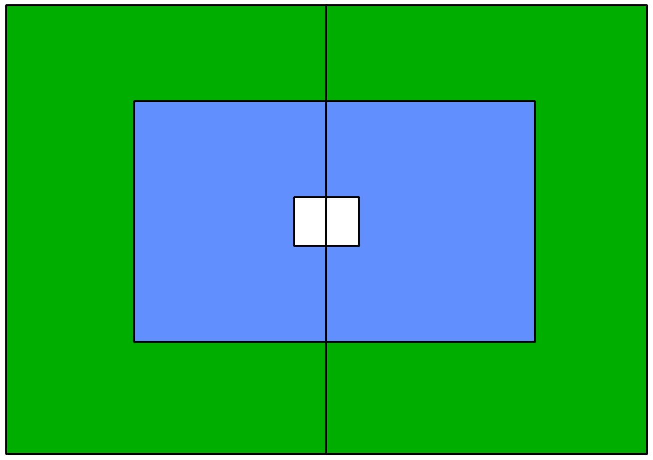
- **This is my own personal views**
  - **This work was performed in December and January, to produce a first version of the Calorimeter Detector description.**
  - **This work hasn't been reviewed by the Calorimeter group**
  - **One of the result was to suggest improvements to the Gaudi team, which have been implemented for the next (= this week's) release.**
- **All mistakes and misunderstandings are mine.**

What is the problem ?

- **Wanted functionality**
  - **Get the position (x,y,z), the transverse size, and the list of neighbours of every calorimeter cell**
- **Approach**
  - **Don't describe each cell in the database !**
  - **Describe the calorimeter as volumes containing cells of the same size, deduce from the big volume the wanted properties.**
  - **All questions addressed to a Calorimeter Detector Element**
  - **Allow a different cell numbering scheme for HCAL and ECAL+PreShower+SPD, keeping the same routines**

Solution for the geometry

- **Outer/Inner**
  - **Cell size is a property of the Detector Element.**
- **Two independent halves**
  - **For alignment.**
- **Need C-shaped volumes**
  - **This is a subtraction of boxes.**
- **Put all that in a big box**
  - **Handle also Z**



Understand the structure

- **DetectorElement**
  - A DetectorElement is an object you can talk to
    - Can have properties, like CodingBit or CellSize
    - Global calorimeter and Sub Calorimeter will be DetectorElements
  - It is a logical volume and has a support, refereed to by a 'support' and a 'rpath'
  - It has a classID to identify that this is a special DetectorElement
    - This ID should match the ID in the C++ header file !

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DDDB SYSTEM "xmldb.dtd" [
  <!ELEMENT CodingBit EMPTY>
  <!ATTLIST CodingBit value CDATA #REQUIRED>
  <!ELEMENT CellSize EMPTY>
  <!ATTLIST CellSize value CDATA #REQUIRED>
]>
<DDDB>
  <detelem classID="8900" name="Ecal">
    <author>Olivier Callot</author>
    <version>0.1</version>
    <geometryinfo logvol  = "/dd/Geometry/lvEcal"
                  support = "/dd/Structure/LHCb"
                  rpath   = "5" />
    <detelemref classID="8901" href="#Outer"/>
    <detelemref classID="8901" href="#Inner"/>
    <specific>
       <CodingBit value="6"/>
    </specific>
  </detelem>

  <detelem classID="8901" name="Outer">
    <author>Olivier Callot</author>
    <version>0.1</version>
    <geometryinfo logvol  = "/dd/Geometry/lvEcalOuter"
                  support = "/dd/Structure/LHCb/Ecal"
                  rpath   = "0" />
    <specific>
       <CellSize value="123.96"/>
    </specific>
  </detelem>
```

◆ Volumes

- A Logical Volume describes the shape
  - Box
  - Subtraction
- It has a name which should be unique, material,...
- It may contain Physical Volumes
  - This is a positioned logical volume inside the current logical volume.
  - It has a name, which should be unique, and position attributes.

```
<!--
  ECAL from 12640 for 845+10 mm  => center at 13067
  Shashlik start at 86 mm from the front, for 435 mm => center off by -124 mm
-->

  <logvol material="Vacuum" name="lvEcal">
    <box sizeX="8000." sizeY="7000." sizeZ="855." name="lvEcalBox"/>
    <physvol name="pvEcalOuter"  logvol="lvEcalOuter"  x="0" y="0" z="-124" />
    <physvol name="pvEcalInner"  logvol="lvEcalInner"  x="0" y="0" z="-124" />
  </logvol>

  <logvol material="Vacuum" name="lvEcalOuter">
    <box sizeX="8000." sizeY="7000." sizeZ="855." name="lvEcalBoxOuter"/>
    <physvol name="pvEcalOuterLeft"  logvol="lvEcalOuterLeft"  x="-1983.36" />
    <physvol name="pvEcalOuterRight" logvol="lvEcalOuterRight" x=" 1983.36" />
  </logvol>

  <logvol material="Vacuum" name="lvEcalInner">
    <box sizeX="2500." sizeY="2000." sizeZ="855." name="lvEcalBoxInner"/>
    <physvol name="pvEcalInnerLeft"  logvol="lvEcalInnerLeft"  x="-619.80" />
    <physvol name="pvEcalInnerRight" logvol="lvEcalInnerRight" x=" 619.80" />
  </logvol>
!-- Outer: 32 * 52 cells of 123.96  - 10 * 16 cells  -->

  <logvol material="Vacuum" name="lvEcalOuterLeft">
    <subtraction name="boxEcalOuterLeft">
      <box sizeX="3966.72" sizeY="6445.92" sizeZ="435"  name="boxEOLMain"/>
      <box sizeX="1239.60" sizeY="1983.36" sizeZ="435"                   x="-1363.56" y=0"
                              z="0" name="boxEOLSubtracted"/>
    </subtraction>
  </logvol>

  <logvol material="Vacuum" name="lvEcalOuterRight">
    <subtraction name="boxEcalOuterRight">
      <box sizeX="3966.72" sizeY="6445.92" sizeZ="435"
                              name="boxEORMain"/>
      <box sizeX="1239.60" sizeY="1983.36" sizeZ="435"                    x= "1363.56" y="0"
                              z="0"  name="boxEORSubtracted"/>
    </subtraction>
  </logvol>
```

Other files

- **General files**
  - **lhcb.xml defines the complete detector.**
    - **The order defines the rpath value. This means that if you change the order in this file, you have to change a value in other files.**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE DDDB SYSTEM "xmldb.dtd">
<DDDB>
        <detelem classID="2" name="LHCb" type="passive">
                <author>Radovan Chytracek</author>
                <version>0.1</version>
                <geometryinfo logvol="/dd/Geometry/lvLHCb"/>
                <detelemref classID="9999" href="vertex.xml#Vertex"/>
                <detelemref classID="2"    href="rich1.xml#RICH1"/>
                <detelemref classID="2"    href="shield.xml#Shield"/>
                <detelemref classID="2"    href="magnet.xml#Magnet"/>
                <detelemref classID="2"    href="tracker.xml#Tracker"/>
                <detelemref classID="2"    href="rich2.xml#RICH2"/>
                <detelemref classID="8900" href="ecal.xml#Ecal"/>
                <detelemref classID="8900" href="hcal.xml#Hcal"/>
                <detelemref classID="2"    href="muon.xml#Muon"/>
        </detelem>
  <logvol name="lvLHCb" material="Vacuum">
        <box name="caveBox" sizeX="50000" sizeY="50000" sizeZ="50000"/>
                  <physvol name="VertexSubsystem" x="0" y="0" z="0"      logvol="lvVertex" />
                  <physvol name="RICH1Subsystem"  x="0" y="0" z="1500"   logvol="lvRICH1" />
                  <physvol name="ShieldSubsystem" x="0" y="0" z="2500"   logvol="lvShield" />
                  <physvol name="MagnetSubsystem" x="0" y="0" z="5000"   logvol="lvMagnet" />
<!-- This is a problem, left out
                <physvol name="TrackerSubsystem" x="0" y="0" z="0"  logvol="/dd/Geometry/lvTracker" />
-->
                  <physvol name="RICH2Subsystem"  x="0" y="0" z="10500"   logvol="lvRICH2" />
                  <physvol name="EcalSubsystem"   x="0" y="0" z="13067"   logvol="lvEcal" />
                  <physvol name="HcalSubsystem"   x="0" y="0" z="14157.5" logvol="lvHcal" />
                  <physvol name="MuonSubsystem"   x="0" y="0" z="16000"   logvol="lvMuon" />
        </logvol>
</DDDB>
```

- catalog.xml defines the list of geometry files
  - This file contains also Material and Structure catalogues

```
<catalog name="Geometry">
      <logvolref href="lhcb.xml#lvLHCb" />
          &vertex_geometry;
          &rich1_geometry;
          &shield_geometry;
          &magnet_geometry;
          &tracker_geometry;
          &rich2_geometry;
          &ecal_geometry;
          &hcal_geometry;
          &muon_geometry;
  </catalog>
```

Per detector

- ecal_geometry.xml lists (references) the various logical volumes which are defined in ecal.xml

```
<logvolref href="ecal.xml#lvEcal" />
<logvolref href="ecal.xml#lvEcalOuter" />
<logvolref href="ecal.xml#lvEcalInner" />
<logvolref href="ecal.xml#lvEcalOuterLeft" />
<logvolref href="ecal.xml#lvEcalOuterRight" />
<logvolref href="ecal.xml#lvEcalInnerLeft" />
<logvolref href="ecal.xml#lvEcalInnerRight" />
```

- Its need is unclear for me now…

C++ Code

◆ Detector Elements

- Need one special Detector Element as soon as one wants some special property. This implies
  - DeCalorimeter.cpp and DeCalorimeter.h with the proper classID

```
const CLID& CLID_DECalorimeter = 8900;  // User defined
```

  - XmlDeCalorimeter.cpp and XmlDeCalorimeter.h to convert the XML file and create the detector element
    - ➨ This file is full of technicalities, only a few lines are specific. See later

- Four files for each new type of detector element
  - We have DeSubCalorimeter just to return the cell size...

```
#include <cstdlib>
#include <string>

#include "Gaudi/Interfaces/ICnvManager.h"
#include "Gaudi/Interfaces/ICnvFactory.h"

#include "Gaudi/Kernel/CnvFactory.h"

#include "Gaudi/MessageSvc/MsgStream.h"

#include "DetDesc/XmlCnvSvc/XmlCnvAttrList.h"
#include "DetDesc/XmlCnvSvc/IXmlCnv.h"

#include "Gaudi/DataSvc/SmartDataPtr.h"
#include "Calo/Xml/XmlCalorimeterCnv.h"
#include "Calo/DetectorElement/DeCalorimeter.h"

extern unsigned char    XML_StorageType;
extern const    CLID&   CLID_DetectorElement;


/// Instantiation of a static factory class used by clients to create
/// instances of this service
static CnvFactory<XmlCalorimeterCnv> calost_factory;
const ICnvFactory& XmlCalorimeterCnvFactory = calost_factory;

const unsigned char& XmlCalorimeterCnv::storageType() {
  return XML_StorageType;
}

/// Report to outside the class ID this converter is used for
const CLID& XmlCalorimeterCnv::classID() {
  return CLID_DECalorimeter;
}

/// Constructor
XmlCalorimeterCnv::XmlCalorimeterCnv(ISvcLocator* svc)
: Converter( XML_StorageType, CLID_DECalorimeter, svc ),
  m_deCnv( 0 ), m_dataObj( 0 ) {

  StatusCode st = serviceLocator()->getService( "DetectorDataSvc",
                                                IID_IDataProviderSvc,
                                                (IInterface*&)m_detSvc);
}

/// Desctructor
XmlCalorimeterCnv::~XmlCalorimeterCnv() {
}
```

```
/// Initialize the converter
StatusCode XmlCalorimeterCnv::initialize()      {

  // Initialize the grand father
  StatusCode status = Converter::initialize();

  ICnvManager*   cnvMgr;

  MsgStream log( messageService(), "XmlCaloCnv" );
  log << MSG::DEBUG << "Initializing calorimeter detector element converter" << endreq;

  if( status.isSuccess() ) {

    status = serviceLocator()->queryInterface(IID_ICnvManager,(void **)&cnvMgr );

    if( status.isSuccess() ) {

      IXmlCnv* xmlCnv;
      const ICnvFactory* cf = cnvMgr->factory( XML_StorageType, CLID_DetectorElement );

      m_deCnv = cf->instantiate( serviceLocator() );

      try {
        xmlCnv = dynamic_cast<IXmlCnv*>(m_deCnv);
      } catch( ... ) {
        log << MSG::FATAL << "Can't get generic detector element converter" << endreq;
        return StatusCode::FAILURE;
      }

      // Must be initialized
      status = m_deCnv->initialize();

      if( status.isSuccess() ) {
        // Register myself as the recevier of User ASCII XML SAX events
         xmlCnv->setUserSaxDocHandler( *this );
      }

      cnvMgr->release();
    }
  }

  return status;
}

/// Finalize the converter
StatusCode XmlCalorimeterCnv::finalize()      {
  // RIP dear grand father!
  return Converter::finalize();
}
```

```
StatusCode XmlCalorimeterCnv::createObj( IOpaqueAddress* pAddress, DataObject*&refpObject) {
  MsgStream log( messageService(), "XmlCaloCnv");
  log << MSG::DEBUG << "Converting ..." << endreq;

  m_dataObj = new DeCalorimeter();

  StatusCode sc = m_deCnv->createObj( pAddress, (DataObject*&)m_dataObj );

  if( sc.isFailure() ) {
    log << MSG::DEBUG << "Failure Converting DeCalorimeter..." << endreq;
    delete m_dataObj;
  } else {
    refpObject = m_dataObj;
  }

  return sc;
}

/// Update the transient object from the other representation.
StatusCode XmlCalorimeterCnv::updateObj( IOpaqueAddress* pAddress,
                                         DataObject* pObject)
{
  return StatusCode::SUCCESS;
}

/// Convert the transient object to the requested representation
StatusCode XmlCalorimeterCnv::createRep( DataObject* pObject,
 IOpaqueAddress*& refpAddress)
{
  return StatusCode::SUCCESS;
}

/// Update the converted representation of a transient object.
StatusCode XmlCalorimeterCnv::updateRep( IOpaqueAddress* pAddress,
                                       DataObject* pObject)
{
  return StatusCode::SUCCESS;
}

StatusCode XmlCalorimeterCnv::fillObjRefs( IOpaqueAddress* pAddress,
                                           DataObject* pObject) {

  return StatusCode::SUCCESS;
}

/// Parsed character data callback
void XmlCalorimeterCnv::uCharacters( const char* const chars,
                                  const unsigned int length ) {
```

```
MsgStream log( messageService(), "XmlCaloCnv" );
  log << MSG::DEBUG << "\"" << chars << "\"" << endreq;

  //if( "stations" == context() ) {
  //  log << MSG::DEBUG << "\"" << chars << "\"" << endreq;
  //}
}

/// White space characters callback
void XmlCalorimeterCnv::uIgnorableWhitespace( const char* const chars,
                                              const unsigned int length ) {
}

/// Start of the XML element callback
void XmlCalorimeterCnv::uStartElement( const char* const name,
                                       XmlCnvAttributeList& attributes) {
  MsgStream log( messageService(), "XmlCaloCnv" );

  std::string tagName( name );

  log << MSG::DEBUG << "<" << tagName << " ";
  for( unsigned int i = 0; i < attributes.getLength(); i++ ) {
    log << MSG::DEBUG << attributes.getName(i)  << "="
      << attributes.getValue(i) << " "
      << attributes.getType(i) << " ";
    ;
  }
  log << ">" << endreq;

  if( tagName == "CodingBit" ) {

  // get a value of the 'value' attribute
    std::string value = attributes.getValue( "value" );

    if( !value.empty() ) {
      log << MSG::DEBUG << "value has value           : " << value << endreq;
      log << MSG::DEBUG << "value has converted value : " << atoi(value.c_str()) << endreq;
      m_dataObj->setCoding( atoi(value.c_str()) );
    }
  }
  else {
// Unknown tag, a warning message can be issued here
  }
}

/// End of the XML element callback
void XmlCalorimeterCnv::uEndElement( const char* const name ) {

  MsgStream log( messageService(), "XmlCaloCnv" );
  log << MSG::DEBUG << "</" << name << ">" << endreq;
}
```

Suggested changes

Reported to the Gaudi team

- **Several minor problems...**
  - **File structure improved. It was more complex before**

- **XML syntax to be improved**
  - **Use of units to specify millimetres, metres, …**
  - **Use of logical units**
    - ➨ Define a cell size
    - ➨ Define the calorimeter size as 64 cells in X and 52 in Y
  - **Default values for non specified quantities**
    - ➨ Do we need to specify x = "0" each time ?

- **Unique names is sometimes a nuisance**
  - **Could be built from the hierarchy**
  - **Not always needed**

- Hardcoding the classID in XML and C++ is no good.
  - Who is allocating the numbers ? One should avoid copies of long code with only minor changes
  - This is a long term maintenance issue
- Referencing files by there order in another file (rpath) is a maintenance issue !
  - The lhcb.xml file defines the rpath to use in ecal.xml
    - Can I add the SPD without affecting ECAL, HCAL, MUON files ?

I had very good support

- My requests were acknowledged, and handled
- I hope the new release will allow an easier use of the XML database, and a simplified DetectorElement C++ code.