Using GaudiObjDesc for Data Model Implementation

Brett Viren

Physics Department



Daya Bay Collaboration Meeting, August 2007

Outline

1. Package Features
2. Alternatives and Benefits
3. How to use
4. Package Status
5. Argument for Mandatory Use
6. Summary

BROOKHAVEN NATIONAL LABORATORY

GaudiObjDesc (GOD) Package Features

- Define data model in simple XML (gdd.dtd format)
- Can use one or multiple XML files in one or multiple packages.
- CMT hooks run GOD scripts:
  - Generate C++ interface headers
  - Generates dictionaries needed to use class in Python or ROOT
- Generates data accessors, optionally turn them off
- Allows for additional C++ implementation code
  - inlined in the XML
  - external, hand written C++ files.

BROOKHAVEN NATIONAL LABORATORY

Alternatives to GaudiObjDesc

Hand written data classes:

- Must write:
  - Definition expressed as C++ header file (`.h`)
  - Additional dictionary XML file (`lcgdict` schema) must list which parts of the definition to be used in ROOT and Python dictionaries.
- Use `reflex_dictionary` CMT pattern to build dictionaries
  - Currently has trouble with `enums`, (maybe due bad failure of C++ parsing?)

What ATLAS uses:

- Don't know.... Anyone?

Note: Use of GaudiObjDesc need not be universal.

BROOKHAVEN NATIONAL LABORATORY

Relative Benefits of GOD vs. `reflex_dictionary`

Pros:

- Encompasses `reflex_dictionary`.
- Single source of definition (one gdd.dtd XML file needed vs. `.h` + `lcgdict` XML)
- Users need to know less about Gaudi's data requirement details.
- XML definition is generic, easier to extend to new uses
- Simple Python processing code is easily hacked.

Cons:

- Structures not in LHCb's model may not be implemented
  - But, enums and free functions added with ∼1 hour work.

BROOKHAVEN NATIONAL LABORATORY

Howto: Write the XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE gdd SYSTEM "gdd.dtd">
<gdd>
  <package name='DataModel' >
    <class name='Hit' id='42' author='bv'
           desc='A deposition of energy' >
      <base name='DataObject'/>
      <attribute type='double' name='energy'
                 desc='Deposited energy'/>
      <attribute type='double' name='time'
                 desc='Time of hit'/>
      <method name='Pe' const='TRUE' type='int'
              desc='Return energy as PE'>
        <code>return (int)(m_energy/100.0);</code>
      </method>
      <method name='PartName' const='TRUE' type='std::string'
              desc='Return particle name'/>
    </class>
  </package>
</gdd>
```

BROOKHAVEN NATIONAL LABORATORY

Resulting generated header file

```
//   ***************************************************************************
//   *                                                                         *
//   *                        ! ! ! A T T E N T I O N ! ! !                    *
//   *                                                                         *
//   *  This file was created automatically by GaudiObjDesc, please do not    *
//   *  delete it or edit it by hand.                                          *
//   *                                                                         *
//   *  If you want to change this file, first change the corresponding        *
//   *  xml-file and rerun the tools from GaudiObjDesc (or run make if you     *
//   *  are using it from inside a Gaudi-package).                             *
//   *                                                                         *
//   ***************************************************************************

#ifndef DataModel_Hit_H
#define DataModel_Hit_H 1

// Include files
#include "GaudiKernel/DataObject.h"
#include "GaudiKernel/boost_allocator.h"
#include <ostream>

// Forward declarations

namespace DayaBay
{

  // Forward declarations

  // Class ID definition
  static const CLID CLID_Hit = 42;
```

```
/** @class Hit Hit.h
 *
 * A deposition of energy
 *
 * @author bv
 * created Tue Jul 31 14:54:53 2007
 *
 */

class Hit: public DataObject
{
public:

  /// Default Constructor
  Hit() : m_energy(0.0),
          m_time(0.0) {}

  /// Default Destructor
  virtual ~Hit() {}

  // Retrieve pointer to class definition structure
  virtual const CLID& clID() const;
  static const CLID& classID();

  /// Fill the ASCII output stream
 virtual std::ostream& fillStream(std::ostream& s) const;

  /// Return energy as PE
  int Pe() const;

  /// Return particle name
  std::string PartName() const;
```

BROOKHAVEN NATIONAL LABORATORY

```
  /// Retrieve const  Deposited energy
  double energy() const;

  /// Update  Deposited energy
  void setEnergy(double value);

  /// Retrieve const  Time of hit
  double time() const;

  /// Update  Time of hit
  void setTime(double value);

  /// operator new
  static void* operator new ( size_t size )
  {
    return ( sizeof(Hit) == size ?
             boost::singleton_pool<Hit, sizeof(Hit)>::malloc() :
             ::operator new(size) );
  }

  /// operator delete
  static void operator delete ( void* p )
  {
    boost::singleton_pool<Hit, sizeof(Hit)>::is_from(p) ?
    boost::singleton_pool<Hit, sizeof(Hit)>::free(p) :
    ::operator delete(p);
  }

private:
  double m_energy; ///< Deposited energy
  double m_time;   ///< Time of hit
}; // class Hit
```

BROOKHAVEN NATIONAL LABORATORY

Minor edits for space. Inlined implementation code not shown.

Howto: Build with GaudiObjDesc

Populate PackageName/xml/ with your XML file(s) and to PackageName/cmt/requirements:

```
# set default namespace (should be in DybPolicy)
macro_append GODflags " -n DayaBay "
# Produce Header-Files from XML description
apply_pattern god_headers files=../xml/*.xml
apply_pattern install_more_includes more=Event
# Generate dictionaries with lcgdict via GOD
macro PackageNameDict_dependencies PackageName
apply_pattern god_dictionary files=../xml/*.xml
# Build implementation library with hand coded C++
library PackageName ../src/*.cpp
apply_pattern linker_library library=PackageName
```

Note: headers will be put in to PackageName/Event/ and installed to InstallArea/include/Event/. Reference them like:

```
#include "Event/Hit.h"
```

BROOKHAVEN NATIONAL LABORATORY

Howto: Use Data Classes in Python

A sampling of how one can access the Hit class in Python.

```
#!/usr/bin/env python
import gaudimodule as gm                # Main module
g = gm.AppMgr()                         # Application manager
g.loaddict('DataModelDict')             # dictionary library
g.Dlls += ['DataModel']                 # hand written code library
g.initialize()                          # Gaudi init
import PyCintex as PC
db = PC.makeNamespace('DayaBay')  # Get top level namespace
print str(db.Hit)              # -> <class '__main__.DayaBay::Hit'>
hit = db.Hit()                          # Create a Hit object
hit.setEnergy(6.9)                      # Use generated accessor
print "Energy is",hit.energy()          # -> Energy is 6.9
```

BROOKHAVEN NATIONAL LABORATORY

Howto: Use Data Classes in ROOT/Cint

```
root [0] Cintex::Enable()
root [1] gSystem->Load("libDataModelDict")
root [2] gSystem->Load("libDataModel")
root [3] DayaBay::Hit* hit = new DayaBay::Hit
root [4] hit->setEnergy(6.9)
root [5] cout << "Energy is " << hit->energy() << endl;
Energy is 6.9
```

BROOKHAVEN NATIONAL LABORATORY

Howto: Use Data as a TTree Branch in Python

```
#!/usr/bin/env python
import gaudimodule as gm                # Gaudi setup
g = gm.AppMgr()
g.loaddict('DataModelDict')
g.Dlls += ['DataModel']
g.initialize()
import PyCintex as PC                   # Access our
db = PC.makeNamespace('DayaBay')        # data class
hit = db.Hit()
from ROOT import TTree, TFile           # Make the tree
f = TFile("hits-P.root","RECREATE")
t = TTree("t","Some Hits")
t.Branch("hit",hit)
import random
for n in range(0,1000):                 # Fill tree
    hit.setEnergy(random.random())
    hit.setTime(n+random.random())
    t.Fill()
t.Write()
f.Close()
```

BROOKHAVEN NATIONAL LABORATORY

Howto: Use Data as a TTree Branch in ROOT/Cint

```
Cintex::Enable();                          // Introduce ROOT
gSystem->Load("libDataModelDict");         // to our data
gSystem->Load("libDataModel");             // classes

f = new TFile("hits-C.root","RECREATE");   // Setup tree
t = new TTree("t","Some Hits");
DayaBay::Hit* hit = new DayaBay::Hit;
t->Branch("hit","DayaBay::Hit",&hit);

TRandom r;
for (int ind=0; ind<1000; ++ind) {         // Fill tree
    hit->setEnergy(r.Rndm());
    hit->setTime(ind+r.Rndm());
    t->Fill();
}
t->Write();
f->Close();
```

BROOKHAVEN NATIONAL LABORATORY

Howto: Read Resulting Tree in ROOT

Resulting TTree from above Howto's will not contain sufficient information to be read back in. Before reading the file you must do:

```
Cintex::Enable();
gSystem->Load("libDataModelDict");
gSystem->Load("libDataModel");
```

This issue needs further looking into.

BROOKHAVEN NATIONAL LABORATORY

Status of GaudiObjDesc Package

- Maintained upstream in LHCb CVS.
  - Older, abandoned version in Gaudi CVS should not be looked at
  - LHCb developers have offered to move back to Gaudi CVS to accommodate Daya Bay
- Mirrored in our SVN at `gaudi/GaudiObjDesc`
  - Just as easy to mirror from LHCb CVS as from Gaudi CVS
  - Local mods possible, while tracking upstream development
- Stable but LHCb eagerly accepts useful patches
  - Non-default namespace, class inherits package namespace
  - Support enum & free function in namespace. Patch needs work so not yet accepted, but LHCb want to include it.
- Receive help from developers and knowledgeable users on `gaudi-talk@lists.bnl.gov` (go to https://lists.bnl.gov/mailman/listinfo/gaudi-talk to sign up)

BROOKHAVEN NATIONAL LABORATORY

Should we mandate the use of GaudiObjDesc for our data classes?

It will greatly reduce our effort by extending it to solve some big problems:

Data object memory managment: maintain a reference count for each object, delete when it hits 0. Entrust wrapper class to inc/dec ref in its ctor/dtor. For convenience, wrapper should contain the methods, data object can just be a struct.

Data I/O mechanism: may require data aware converters or marshallers.

Database table creation: DBI objects have a one-to-one corelation between database columns and class data. Nice to have SQL to initialize the tables in the database for new classes.

- GaudiObjDesc can be easily extended to emit the required code.
- People not using GOD will have to write this code by hand.

BROOKHAVEN NATIONAL LABORATORY

Summary

- GaudiObjDesc provides a simple but powerful way to implement our data model classes.
- It generates Python and ROOT dictionaries for you.
- LHCb provides help and is accepting of patches.
- Source mirrored in our SVN.
- I recommend GaudiObjDesc used for all our data classes.

More info in this wiki topic[1]:

https://wiki.bnl.gov/dayabay/index.php?title=GaudiObjDesc

[1] This is currently Google's "I'm feeling lucky" link when searching on "GaudiObjDesc"!

BROOKHAVEN NATIONAL LABORATORY