An Introduction to Offline Software

Erik Blaufuss
University of Maryland

Reconstruction/Offline Software Workshop
Uppsala Collaboration Meeting
October 9, 2004
What is Offline Software?

• Software and tools used to provide an analysis framework for simulation, reconstruction and analysis.

• This includes:
  – IceTray Software Framework
  – DataClasses Storage classes
  – Input/Output utility modules
  – 3-D Event Viewer
  – Services (DB, random numbers, logging, etc)
  – User supplied modules (YOUR CODE HERE.)
The Philosophy

- Provide a single, easy to use framework for analysis of IceCube data
- New analysis and reconstruction modules can move transparently between:
  - Development work on a physicist’s desk
  - Testing with simulation data
  - Production processing at Data Warehouse
  - South Pole reconstruction/filtering
- Provide many of the support and utility modules needed for data reco/analysis.
- Easy to combine modules to make new apps
  - Clean, easy to understand module interface and steering
An overview of today…

• Part 1: Introduction and demo
  – Introduction to offline software and components
  – Tools used to create offline software
  – Interactive demonstration

• Part 2: Information for module developers
  – An introduction to the programming techniques and ideas used in IceCube software
  – A guide for writing and understanding how a module works
  – Services and other information you’ll need
  – Lessons learned along the way.
An overview of today (2)

- **1st** release of offline software is available
  - V01-00-00 of DATACLASS-APP
- Much of what I will talk about and show you is included in this release
  - Some things are from the latest development
- Our goal is to release early and often as new features are added and bugs are fixed
  - V01-01-00 expected soon.
    - Contains a few bug fixes and new functionality.
  - New releases announced on icecube-s mail list.
IceTray Framework

• We’ve chosen a software framework for:
  – Modularity – Weak coupling of modules.
    • Promotes the use of well defined interfaces.
  – Reusability – Modules can be reused as needed
    • Can make an executable by collecting existing modules
  – Extensible – Can easily extend functionality
    • Write only a single module, and add to others.
  – Control, ease and reliability – framework is main()
    • Users only concentrate on module, framework worries about getting all little pieces together to build an executable.
Data Model

• Using the Frame-Stream-Stop model
  – “Full” picture of detector made of objects that change on different time scales.
  – Break picture up into different streams, and a “stop” on a stream generated when it changes.
  – Modules register interests in particular stops
    • Most likely, the Physics (events) stop most interesting.
IceTray Modules

- Physicists supply IceTray modules to perform reconstructions and analysis.
- How it works:
  - Frames appear (Inbox) for you to operate on
    - Perform reconstruction, analysis, etc based on stops.
    - Add results to appropriate place (Dataclasses)
    - Drop in appropriate Outbox
  - Each module has set of Inbox(s)/Outbox(s)
    - Flow of application is defined by connecting outboxes and inboxes appropriately.
    - Steering of modules defined in steering file
Dataclasses

• The set of objects that are used by the framework to hold IceCube specific data.
  – Signals, waveforms readout from PMTS
  – Trigger,
  – Monte Carlo monitoring, geometry information
  – Results of Reconstructions information
  – History information

• Visible part for end user, very important for developers of modules for framework
  – “Language” used by modules to communicate
Dataclass Details

All the standard C++ / Root tools and ideas:

• Well defined structure for a Frame
  – Each stream is defined in Dataclasses
  – Event, Geometry well defined.
  – Some still need work (Monitoring, status information)
  – Each entry in Frame has header for DB access

• Use methods to provide access to data
  – Get/Set methods
  – Can re-implement internal representation

• A completely modular design
  – Policy classes define storage containers and pointer types
  – Can be changed as needed, with little or no effect on user code
Dataclass Details (2)

• Root automatic schema evolution
  – Classes evolve, Root will read old files into new versions of classes

• STL Containers Classes
  – Vectors, Maps etc used to store Dataclass objects.
  – Can be heterogeneous.
    • Allows for storage of different, but similar objects in same container

• Base classes are provided
  – Tracks, hits, OM status, geometry, etc.
  – Users are expected to extend these to customize code
Frames Overview

I3Frame
I3Frame is part of IceTray, not the dataclasses project
- GetElement()
- LendElement(Element, Name)
- PutElement(Element, Name)
  These functions allow arbitrary named streams -- only some shown here

I3EventHeader
- GetJulianDay()
- GetSec()
- GetNanoSec()
- GetRunID()
- GetEventID()
- GetDataStream()
  DataStream() returns the stream name

I3Event
- GetTriggerDict()
- GetFilterDict()
- GetOMResponseMap()
- GetTopResponseDict()
- GetRecoResultDict()
- GetBag()

I3DetectorStatus
Contains detector settings
- GetJulianDay()
- GetSec()
- GetNanoSec()
- GetDataStream()

I3DetectorStatusHeader
- GetJulianDay()
- GetSec()
- GetNanoSec()
- GetDataStream()

I3MonitoringHeader
- GetJulianDay()
- GetSec()
- GetNanoSec()
- GetDataStream()

I3Monitoring
Contains measured info -- more to come
- GetAtmosphere()

I3GeometryHeader
- GetJulianDay()
- GetSec()
- GetNanoSec()
- GetNumOMs()
- GetNumStrings()
- GetArrayName()
- GetDataStream()
Event Overview

**I3Event and I3EventHeader**
- GetJulianDay()
- GetSec()
- GetNanoSec()
- GetRunID()
- GetEventID()
- GetTriggerDict()
- GetFilterDict()
- GetOMResponseMap()
- GetTopResponseDict()
- GetRecoResultDict()
- GetBag()
- GetDataStream()

**I3OMResponseMap**
- Map<I3OMResponsePtr>

**I3TopResponseDict**
- Map<I3ArrayHitPtr>

**I3RecoResultDict**
- Map<I3RecoResultPtr>

**I3TriggerDict**
- Map<I3TriggerPtr>

**I3FilterDict**
- Placeholder until details are available
- Map<I3FilterPtr>

**I3Bag**
- Generic TObject container
- Map<TObjectPtr>
Inheritance

- Extension through inheritance
  - `I3BasicTrack → I3DipoleFitTrack`
  - `I3BasicTrack` defines and implements the minimal set of Track information (vertex, direction, length…)
  - `I3DipoleFitTrack` inherits the basics, then adds on the information and functionality specific to the Dipole Fit algorithm (dipole moment)
- Clients interested only in the basics ($\theta, \varphi$) need use only the `I3BasicTrack` interface
- Clients who want the specific info can access the specialized parameters through casting
- More details in Ty’s talk later.
Input/Output Modules

- Read F2K data format (Uses rdmc libraries)
  - Includes TWR waveforms as well
- Reads/writes ROOT I/O format files
  - Based on TTree structure of root, one Tree for each type of data stream
  - An example stores the order in which physics events (P), geometry events (G), calibration events (C), etc. occur:
    - G C P P P C P C P P P G P P P P.
  - The objects itself are put into branches
    - G G
    - C C C
    - P P P P P P P P P P P P
Other pieces

- Event Viewer
  - A ROOT based, 3-D event viewer
  - Still in development, but close to release
  - Demonstration today.

- Services
  - A collection of tools to be used by all modules
    - PhysicsModule
    - Random number generators.
    - Particle Data Service
    - Calculator Service for track geometry quantities
    - Database access
Tool set

- In addition to the code, there are a set of software tools to provide:
  - Proper code compilation
  - Version tracking
  - Testing
  - Documentation.
- A talk on many of these is next.
Bugs

• My one guarantee:
  – *There are bugs in this software.*

• When you find them, please let us know!
  – The Gateway to our bug tracking system:
    – offlinebug@icecube.umd.edu
    – Please include as much detail as possible!

• General questions, comments and requests
  – Email the developer’s email list:
    – dataclass@icecube.wisc.edu
Documentation

• Using the DOXYGEN documentation system
  – Automatically generates Class documentation
  – Shows inheritance well
• Additionally, developer written overview and supplemental documents are added into DOXYGEN system
  – Includes step by step “Getting Started” guide
• All collected in one place:
  – http://glacier.lbl.gov/ offline
How to get started…

• The ONE thing to take away from this talk:

http://glacier.lbl.gov/offline

• Your one stop link collection for all IceCube Offline Software.
  – Introduction and Overview
  – Help getting started, compiling
  – Details on software packages
  – Updated often.