Introduction to Podd

Hall A Data Analysis Workshop
06 January 2007

Ole Hansen
Jefferson Lab

Podd – The C++ Analyzer

- Standard Hall A analysis software.
- Built on top of ROOT as an interactive ROOT application.
- Toolbox of analysis modules: apparatuses, detectors, physics modules, etc. "Everything is a plug-in".
- Tests/Cuts and Output can be defined dynamically (no recompilation).
- Time-dependent database (text files or SQL).
- **Requires good knowledge of C++**

Podd Concepts

AnalysisObjects:

- Detector
- Apparatus (collection of detectors, analyzed together)
- Physics Module (may combine data from several apparatuses)
- "Analysis Variables": Results exported by any of the above

Miscellaneous

- "Analyzer" (event loop)
- Test/Cut definitions (analysis control)
- Output definitions (tree variables, histograms)
- "Run" objects (description of input source)
- Decoder (interface to raw data, CODA or other)

Analysis Objects

- Common base class for Apparatus/Spectrometer, Detector, Physics Module.
- Defines common interface (API).
- Provides support for naming objects, debugging, global variables, database access and similar.
- Name is important — unique identifier!

Analysis Variables

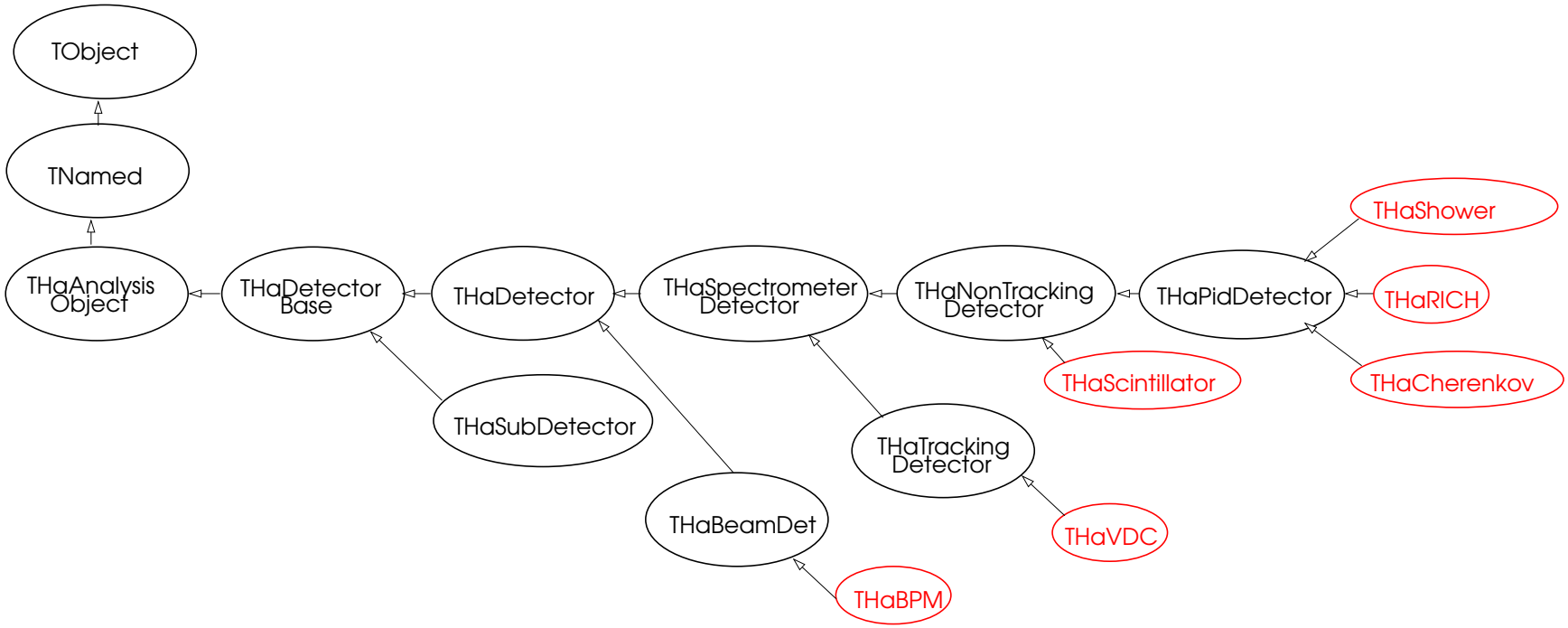
(a.k.a. "global variables")

- Contain results exported from any AnalysisObject.
- Scalers, fixed and variable arrays. Any numerical data type.
- Named, prefixed by Apparatus.Detector names. Example: "L.vdc.u1.nhits".
- Read by test/cut and output packages.
- FAQ: Which variables are available?
  `http://hallaweb.jlab.org/root/doc/variables.html`
- Negligible performance penalty for adding more variables.

Detector Classes

- Code for analyzing a *type* of detector. Examples: Scintillator, Cherenkov, VDC, BPM.
- Generic design: One class describes any number of similar detectors, differing only by name and database entries.
- Embedded in Apparatus or standalone.
- Must not assume presence of other detectors.
- Must provide a Decode() function.

Detector Class Hierarchy



(Podd 1.4)

Apparatus/Spectrometer Classes

- Collection of Detectors.
- Examples: HRS, BigBite, ND, Beam
- May combine decoded data from detectors.
- A "Spectrometer" is an Apparatus with support for tracks and a standard Reconstruct() function.
- Must provide Decode() and Reconstruct() functions.

Physics Modules

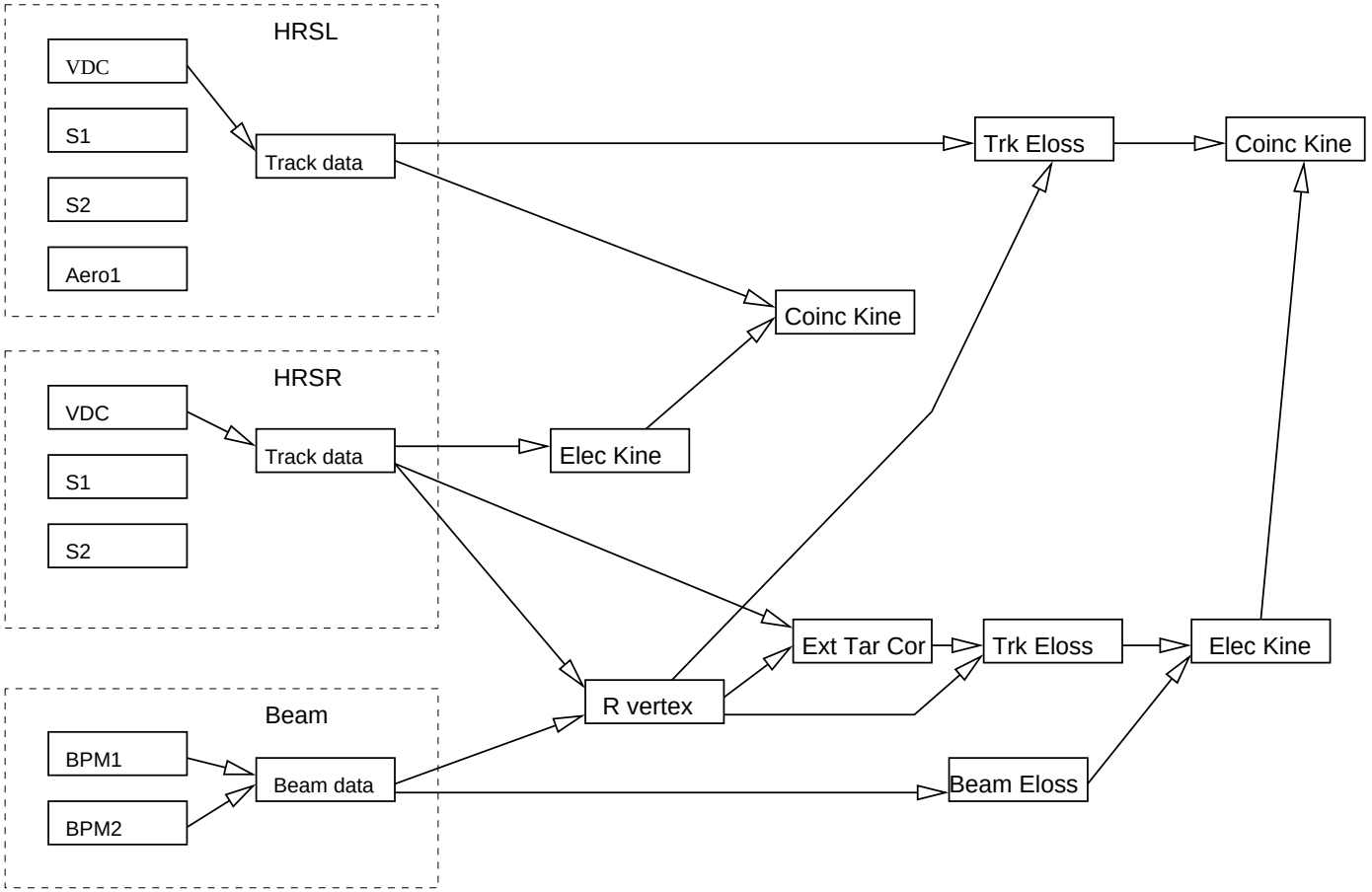
- May combine data from several apparatuses.
- Typical applications: kinematics calculations, vertex finding, corrections, coincidence time, statistics
- Special applications: debugging, event display, custom output
- Any other event-by-event usercode.
  (But: direct analysis of hardware data should be done by a detector.)
- "Plug-in" design. Use & combine as needed.
- Output from modules that provide tracking info can be used as input for modules that operate on tracking info, etc.
- Modules may export variables, define histograms, etc., that can be written to output ROOT file.
- Must provide Process() function.

Current Physics Modules

Kinematics	Single-arm	THaPrimaryKine
		THaElectronKine
	Coincidence	THaSecondaryKine
	Photoproduction	–to do– (?)
Vertex	Single-arm	THaReactionPoint
	Two-arm	THaTwoarmVertex
		THaAvgVertex
Corrections	Extended target	THaExtTarCor
	Energy loss	THaBeamEloss
		THaTrackEloss
Timing	HRS coincidence	THaCoincidenceTime
Special	Debugging	THaDebugModule
	Golden Track	THaGoldenTrack
	Event Display	–prototype–

Current limitation: Only operate on Golden Track

Analysis Module Configuration Example



Event loop

- THaAnalyzer.
- Manages lists of objects, output, cuts, decoder, file names, etc.
- Handles initialization.
- Not required. Can create custom event loop, even in script.
- G_E^n experiment has developed an extension with more flexible analysis flow control ($\longrightarrow$ Rob's talk).

Tests & Cuts

http://hallaweb.jlab.org/root/doc/test-guide.html

Blocks of tests defined in THaAnalyzer:

- RawDecode: After THaEvData::LoadEvent().
- Decode: After Decode().
- CoarseTracking: After CoarseTrack().
- CoarseReconstruct: After CoarseProcess().
- Tracking: After FineTrack().
- Reconstruct: After FineProcess().
- Physics: After Process() of all physics modules, *i.e.* right before writing the output.

```
# --------  Example e12345.cuts  --------------------------
Block: RawDecode
evtyp1            g.evtyp==1          // Event type 1 (=HRSR main trigger)
poshel            g.helicity==1
neghel            g.helicity==-1
goodhel           poshel||neghel
RawDecode_master  evtyp1

Block: Decode
NoisyU1           R.vdc.u1.nhit>50
NoisyV1           R.vdc.v1.nhit>50
NoisyU2           R.vdc.u2.nhit>50
NoisyV2           R.vdc.v2.nhit>50
NoisyVDC          NoisyU1||NoisyV1||NoisyU2||NoisyV2
EnoughShowerHits  R.sh.nhit>10
Decode_master     !NoisyVDC
```

Output

http://hallaweb.jlab.org/root/doc/output.html

```
# --------  Example e12345.odef  -------------------------

# Variables to appear in the tree.
variable L.s1.lt[4]
variable L.s1.rt

# The 'block' variables:  All data in Right HRS go to tree.
block R.*

# Formulas can be scalers or vectors.
# Lt4a is a scaler.
formula Lt4a 5.*L.s1.lt[4]

# Cuts can be defined globally and used in histograms.
# Cut C1 is a scaler.  Data is 0 or 1.
cut C1 L.s1.lt[4]>1350

# Histograms can involve formulas, variables, and cuts. TH1F, TH1D, TH2F, TH2D supported.
TH1F  rv1n  'L-arm vdc hits on V1'  L.vdc.v1.nhit  10 0 10
```

Decoder

- THaEvData (abstract base class), THaCodaDecoder.
- Decodes raw CODA event buffer into crate/slot/channel data.
- Currently supports the Fastbus & VME modules used in Hall A.
- Supporting new hardware currently requires expert.
- Abstract interface allows easy extension to other input formats (e.g. Monte Carlo data).

Run objects

- THaRun (CODA file), THaOnlRun (ET data).
- Stores run info, e.g. raw data file name, time, run number, event number ranges, etc.
- Stores global run parameters, e.g. beam energy, target mass, etc., obtained from database.
- Now supports split CODA files correctly.

Example Script

```
gHaApps->Add( new THaHRS("R", "Right HRS") );
myrun = new THaRun("/data/e12345_6789.dat");
analyzer = new THaAnalyzer;
analyzer->SetOutFile("/work/e12345_6789.root")
analyzer->Process(myrun)
// now inspect results in memory, or quit
```

Requires:

- Database for HRS-R detectors
- output.def (default output definition file)

Debugging

If Things Don't Work:

- Inspect tree output using `tree2ascii` tool.
- Inspect analysis variables using `THaDebugModule`.
- Pay attention to "mysterious" messages.
- Call `SetDebug()` to get more verbose messages.
- Advanced: Recompile Podd & your library with `DEBUG=1`, link against a debug version of ROOT, and run under `gdb`.

Extending Podd

The core code is a fixed library that users do not modify. Extensions are incorporated through external libraries dynamically loaded at run time.

If new modules (detectors, apparatuses, physics modules) are needed:

- Write the code for the module(s) and compile it into a shared library
  Hint: Use our "Software Development Kit" (SDK) to get started.
  - Take advantage of inheritance
  - Use utility functions provided in core library (e.g. for database access)
- Create the database(s) for the module(s)

Once experiment-specific library is written, using it is a simple matter:

```
gSystem->Load("libKaon.so");
HRSL->AddDetector( new THaRICH("rich", "The Hall A RICH" ));
```

Recent Developments

- Version 1.4. Many small improvements. Release Notes on the Web.
- GEn library (BigBite, neutron detector).
- FPP code.
- New kinematics modules (photoproduction, . . .).

Work in progress

Nearly done: (for Version 1.5)

- New database interface (`THaDB`)
- Cleaned-up helicity code.

Longer-term

- BigBite tracking

Open Issues

- **User's Guide**
- Output performance
- VDC issues
  - Improve (or: better quantify) tracking efficiency
  - Calculate t_0 of cluster fits and use it for cluster matching
  - Exploit multi-hit capability of TDCs to remove noise
- Shower cluster algorithm
- Misc. code cleanup
- Tools (GUI, browser, DB editor, event display . . .)

Further Reading

- Your favorite C++ book
- Podd documentation at:

  `http://hallaweb.jlab.org/root/`

- ROOT User's Guide at:

  `http://root.cern.ch/`

- Example scripts in $ANALYZER/examples
- Online analysis setup for recent experiments (adaq account)