Analysis Framework Overview & Progress

Ole Hansen

Jefferson Lab

Hall A & Hall C Data Analysis Workshop
January 14, 2015

Hall A Analysis Framework ("C++ Analyzer", "Podd")

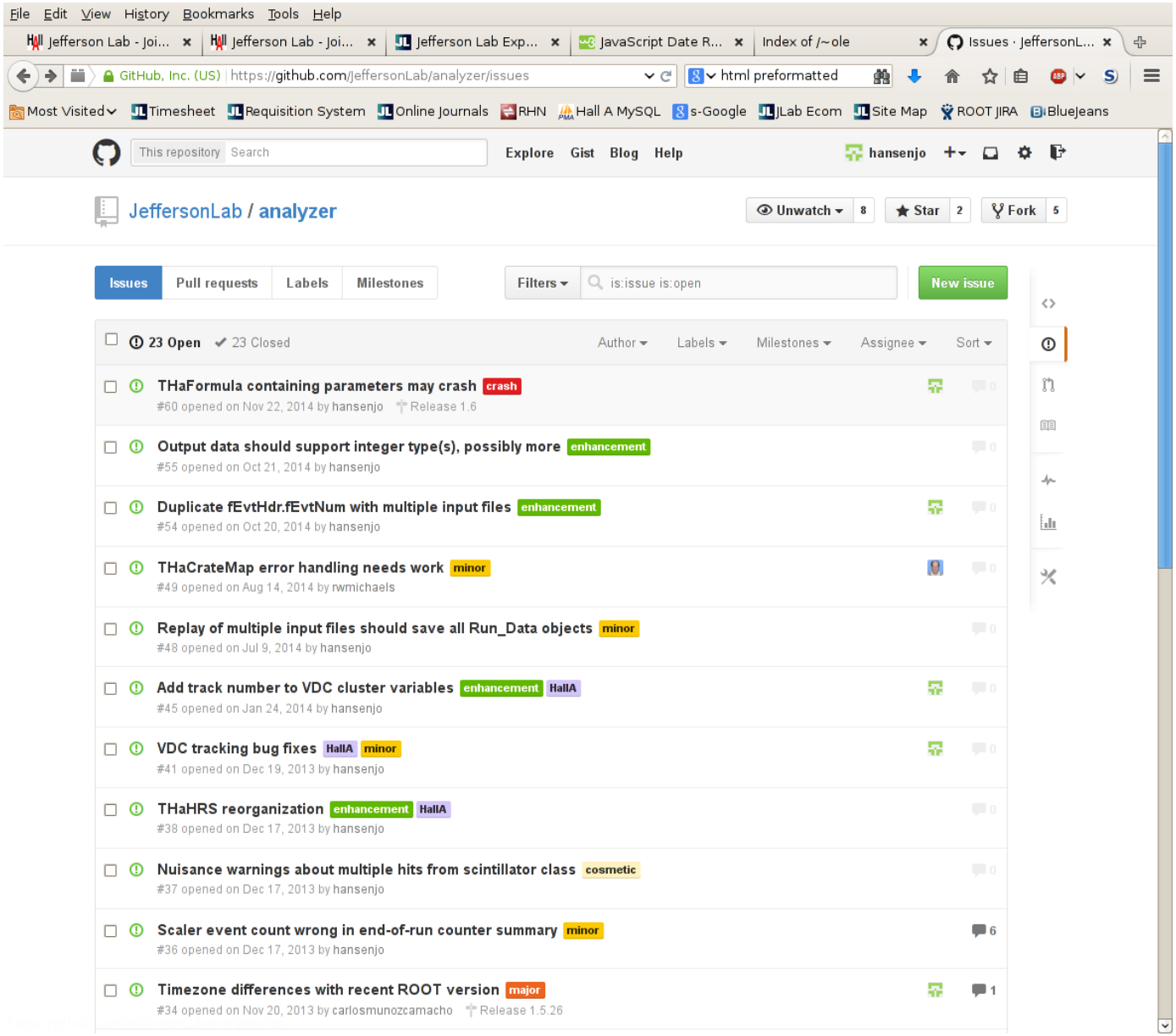
- Class library on top of ROOT
- Highly modular: "Everything is a plug-in"
- In production use since 2003
- Modules for all 6 GeV-era Hall A equipment exist and are well tested
- Software for non-standard equipment typically provided by users ("experiment library"). SDK available for rapid development.
- Currently being adopted by Hall C ▸ Steve Wood's talk

Current Version: Podd 1.5.28

Version 1.5 is in maintenance mode, *i.e.* we only apply binary-compatible bugfixes & small improvements

- Updates in 2014
  - Improved CAEN 1190 decoder (Brad Sawatzky)
  - Tweaks for Hall C (Steve Wood)
  - Updated SDK
  - Assorted bugfixes
  - Support for latest ROOT, Linux and compiler versions (ROOT 6 not yet supported)
- Download/documentation: `https://hallaweb.jlab.org/podd/`
- Source code repository: `https://github.com/JeffersonLab/analyzer.git`
- Bug tracking using GitHub's issue tracker. Anyone welcome to report issues.

Github Issue Tracker



Next Release: Podd 1.6

- Object-oriented decoder ✓ ▸ Bob Michaels's talk
- VDC bugfixes ✓
- Simulation decoder framework ✓
- Improved tests & formulas ✓
- `std::vector` global variables ✓
- Object-oriented `THaDecData` & `VDCeff` modules ✓
- Abstract database interface ✓
- Updated SDK
- Time-zone safe `TTimeStamp` for time stamps
- Test & validation procedures ✓
- EVIO from external library ✓
- Code split into core and hall-specific libraries

✓ done, ✓ partly done

ETA: Spring 2015

VDC Analysis Software Status & Progress

A thorough code review in late 2013 revealed several bugs in the handling of multi-cluster VDC events (typ. $< 10\%$ of total). We concluded:

> The current VDC tracking algorithm is definitely broken for events with multiple clusters in more than one plane. Such events should be rejected in any analysis using the present code.

Details: `https://userweb.jlab.org/~ole/HRS-Tracking-HallAMtg.pdf`

Since then

- Bugfixes included in Release 1.6
- Extensive additional VDC analysis code has been developed (improved cluster fitting, cluster splitting, global optimization) About 80% finished. To be tested.

Generic Simulation Decoder Framework

- Derived decoder class `Podd::SimDecoder` (in `SimDecoder.h`), designed for simulation data input
- Provides generic access to MC truth data (tracks, hits), directly and via global variables
- Allows detailed studies of reconstruction performance, reasons for reconstruction failures, background contamination of measured signals etc. Extensively used in SoLID tracking simulations.
- Doesn't come for free: user must implement actual interface to simulation data & fill data structures. Reconstruction code usually needs to be slightly modified as well.
- Included in Release 1.6

Improved Global Variables, Formulas & Tests

std::vector global variables

```
UserDetector.h:
  std::vector<float>  fData;

UserDetector::DefineVariables( EMode mode ) {
  RVarDef vars[] = {
    { "data", "Data values", "fData" }, // var-size array of floats
    { 0 }
  };
  return DefineVarsFromList( vars, mode );
};
```

Array formulas

```
vector<int> vi;
vi.push_back(100); vi.push_back(200); vi.push_back(300); vi.push_back(400);
gHaVars->Define("vi",vi);

THaFormula* f1 = new THaFormula("f1","vi+vi/10"); // Variable-size array formula
f1->IsVarArray()
1
f1->EvalInstance(3)
440
f1->EvalInstance(4)
(Double_t)9.9999999999999977e+37
f1->IsInvalid()
1
```

Improved Global Variables, Formulas & Tests II

Cuts defined on variable-size arrays

```
SimDecoder.h:
  class Podd::MCTrack : public TObject {
    UInt_t   fHitBits;  // Bitpattern of plane numbers with hits used by this track
    ...
  };

UserTracker.h:
  TClonesArray*   fTracks;  // holds Podd::MCTrack objects

UserTracker::DefineVariables( EMode mode ) {
  RVarDef vars[] = {
    { "tr.planebits", "Plane hit bits", "fTracks.Podd::MCTrack.fHitBits" },
    { 0 }
  };
  ...
};

replay.cuts:
  Block: RawDecode

  OneTrackAllPlanes    MC.tr.n==1&&MC.tr.planebits[0]==0xFF
  AnyTrackAllPlanes    MC.tr.planebits==0xFF     # Implicit OR
  AllTracksAllPlanes   AND:MC.tr.planebits==0xFF
  RawDecode_master     AnyTrackAllPlanes
```

Rewritten THaDecData Generic Decoder

- Object-oriented design. New channel types can be added via plug-ins.
- Use module as before
  - Same constructor
  - Same decoding functionality
- Optional new database format (see box at right)
- VDC efficiency calculation moved to standalone `VDCeff` module

New THaDecData database format

```
# Version: 2

D.word =
# syncclock
#      varname     crate  header       data offset past hdr
       syncroc1    1      fabc0006     6
       syncroc2    2      fabc0006     6
       syncroc3    3      fabc0008     8
       syncroc4    4      fabc0008     8
# Deadtime Calculations
       R1start     1      0xdeaddead   1
       R1stop      1      0xdeaddead   2

D.multi =
# trigger inputs (1877) left arm copy
#      varname          crate  slot     channel
       lt1              4      23       64
       lt2              4      23       65
# Electronic deadtime
       edtpl            4      23       74
       edtpr            2      11       58
# Coincidence time
       ctimel           4      23       72
       ctimer           2      11       56
```

Database API

- Retain 1.5 API in v1.6+ for backward compatibility
- Only minimal code changes required (see code snippets)
- v1.6+ API allows different backends, *e.g.*
  - Hall A-style flat files
  - Hall C-style parameter file
  - MySQL server
- Backend can be set and/or configured from replay script

Podd 1.5 Database Access

```
UserDetector::ReadDatabase( const TDatime& date ) {
  FILE* file = OpenFile( date );
  DBRequest request[] = {
    { "planeconfig", &planeconfig, kString },
    { "MCdata",      &mc_data,     kInt,    0, 1 },
    { 0 }
  };
  Int_t err = LoadDB( file, date, request, fPrefix );
  fclose(file);
};
```

Podd 1.6+ Database Access

```
THaInterface.C:
  THaDB* gHaDB = new THaFileDB( DB_DIR );  // Default DB

UserDetector::ReadDatabase( const TTimeStamp& date ) {
  DBRequest request[] = {
    { "planeconfig", &planeconfig, kString },
    { "MCdata",      &mc_data,     kInt,    0, 1 },
    { 0 }
  };
  Int_t err = LoadDB( date, request, fPrefix );
};
```

Assorted Improvements

Timezone-safe time stamps

```
THaAnalysisObject.h:
  EStatus  Init( const TDatime& run_time );
  ...
changes to:
  EStatus  Init( const TTimeStamp& run_time );
  ...
```

Generic Unit Test API

```
namespace Podd {
namespace Tests {

class UnitTest : public THaAnalysisObject {
public:
  UnitTest( const char* name, const char* description );

  virtual Int_t Test() = 0;
  ...
};
```

Open Issues/Tasks

- Core Analyzer
  - Decoders for pipelined readout modules: missing
  - Parallel processing on multi-core systems: not directly supported
  - Scalability to very large data sets: to be tested
- Specific experiments
  - G_M^p: FPP tracker plane (optional, see yesterday's G_M^p talk)
  - APEX: High-rate VDC track reconstruction
  - SBS: GEM tracker & calorimeter reconstruction
  - SBS GEp(5): Recoil polarimetery, kinematic correlation analysis
  - SBS SIDIS: RICH analysis & PID
  - Møller, SoLID: to be determined

Reconstruction Software Status & Tasks

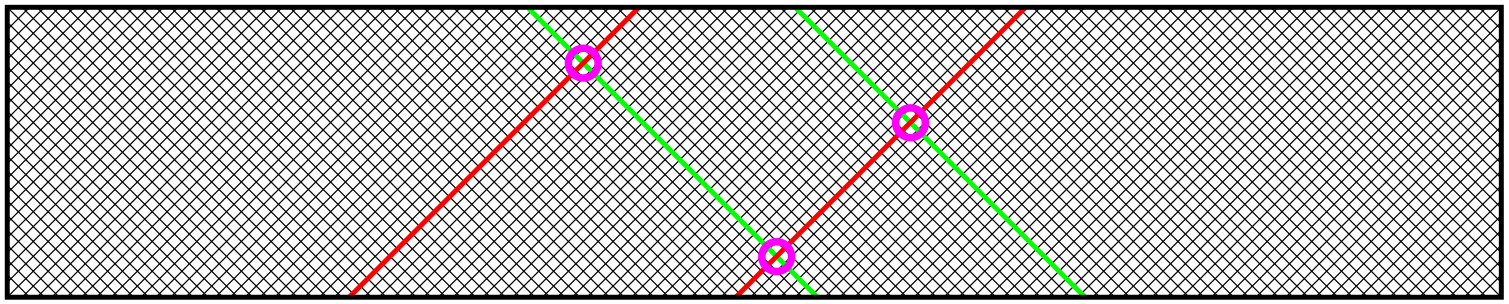
Experiment	Base Software	Required Extensions	Status / Required By
GMp	C++ Analyzer	(FPP tracker integration)	Spring 2015
DVCS	C++ Analyzer	Photon detector analysis	Done
^{3}H/^{3}He	C++ Analyzer	BigBite MWDC track reconstruction	Done
A_1^n	C++ Analyzer	-	-
APEX	C++ Analyzer	High-rate VDC track reconstruction	Spring 2016
PREX/CREX	PAN & C++ Analyzer	-	-
SBS general	C++ Analyzer	• Analyzer parallelization • Pipelined electronics decoder • GEM track reconstruction • BigBite GEM/MWDC tracking • Calorimeter cluster reconstruction	≥ Fall 2017
SBS GEp(5)	C++ Analyzer	• Recoil polarimetry • Kinematic correlation analysis	≥ 2018?
SBS SIDIS	C++ Analyzer	• RICH analysis & PID	≥ late 2018?

Red: not yet written Purple: exists, but incomplete and/or not yet fully tested/integrated

Analyzer Parallelization

- Goal: provide automatic event-level parallelization of any replay
- Ideally, should be as transparent as possible to user code, *i.e.* no or only minimal code modifications necessary
- Initial code review shows that
  - Multi-threading of existing `THaAnalyzer` is possible and relatively easy
  - Required user code modifications:
    - ★ Replace globals like `gHaVars` with per-thread static members like `THaAnalysisObject::fgVars`
    - ★ Provide some form of virtual copy method (to be specified)
  - Most of the work will have to be done in the output module `THaOutput`. Could become a bottleneck due to slowness of ROOT file output.
- Significant project. Need about 3 months of (preferably uninterrupted) time
- ETA: hopefully in 2015

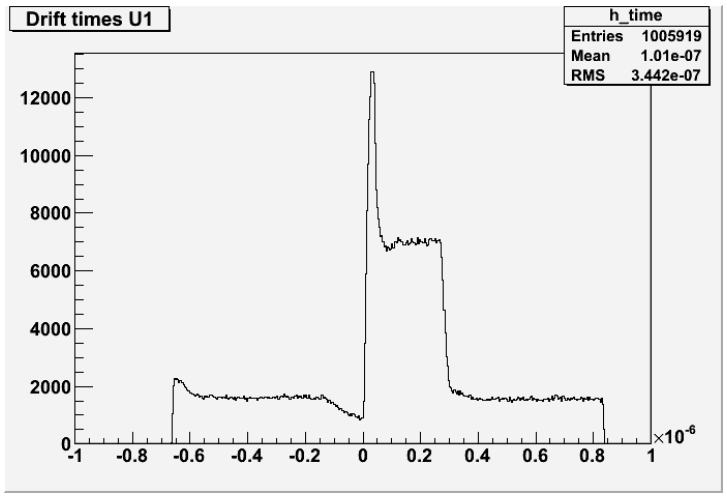
VDC Tracking Improvements I: Multi-Cluster Events

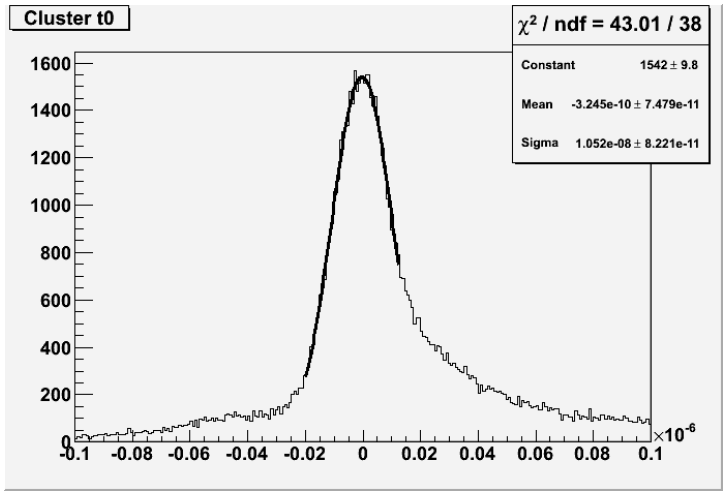


- At low rates, multiple clusters with $t_0 \approx 0$ may occur, often in close proximity to each other
- Long-standing problem. Probably caused by delta electrons
- Causes u-v matching ambiguity
- Cannot be fully resolved in software only, using only VDC data
- 3rd wire direction (expensive) or 3rd tracker plane (less effective) may help $\longrightarrow$ upcoming G_M^p experiment will test
- Fortunately, typically less than 10% of events are affected

VDC Tracking Improvements II: Accidental Coincidences

- Accidentals occur at high singles rates (MHz)
- Cause multiple cluster topology as in previous case
- Can be largely resolved in software only by performing non-linear 3-parameter fit to cluster TDC values to extract track time offset t_0
- $\approx \times 10$–20 background rejection with APEX test data
- Could likely be further improved with 3rd tracker plane, as in previous case
- Prototype analysis code written in 2010, improved version in 2014. To be tested & integrated.





SBS Software

- Beginning to take shape
- Reconstruction software development started
  - Track reconstruction based on neural network algorithm & Kalman filter (INFN)
  - Calorimeter cluster finding (UConn)
  - RICH analysis (UConn)
- Simulations ongoing (Seamus Riordan coordinating)
  - Meetings every two weeks
  - Active GitHub repository
  - Need to develop common interfaces, database
  - Digitization to be done

Conclusions

- Core Hall A & C analysis software is largely ready for 2015–2016 running, but will require upgrades for later 12 GeV experiments
- Work has been ongoing throughout 2014
- Additional manpower very desirable to keep schedule!
- Release 1.6 planned for spring 2015