Introduction to LArSoft

Erica Snider

Fermilab

LArSoft core support team:
Vito di Benedetto, Lynn Garren
Patrick Gartung, Marc Mengel
Gianluca Petrillo, Vladimir Podstavkov,
Saba Sehrish, Erica Snider

art/LArSoft Course
August 3—7, 2015
Fermilab
Goals for this session

- Provide overview of LArSoft project, collaboration and software
- Introduce basic concepts and workflows
- Summarize primary LArSoft design principles
- Provide simple example of how to use LArSoft
- Summarize available resources and documentation
Outline

- What is LArSoft?
- Operation of single-phase LAr TPC
- Primary reconstruction overview
  - Workflow and data structures
- Secondary reconstruction
- Simulation overview
- LArSoft design principles
- Using LArSoft
- Resources
What is LArSoft?

• A project / collaboration
  - Provide an integrated, art-based, experiment-agnostic set of software tools for LAr neutrino experiments to perform simulation, reconstruction and analysis
    • The core LArSoft (“project”) team maintains infrastructure, architecture, interfaces, coordination, code management and distribution
    • Experiment partners provide technical requirements, development effort and coordination, required experiment-specific plug-ins and configuration

• A body of code
  - Core LArSoft products
    • Experiment-agnostic data structures, algorithms, interfaces, etc
    • Lives in a set of repositories managed by the core LArSoft team
  - Experiment-specific components
    • Detector-specific geometry descriptions, electronics response functions, calibration functions, etc.
    • Live in repositories managed by the experiments
What is LArSoft?

- The collaboration of experiments, Fermilab, other stakeholders in the project
  - DUNE/35T/LBNF
  - MicroBooNE
  - SBND
  - LArIAT
  - ArgoNeuT
  - Future participating experiments, laboratories and projects
  - NuTools
  - art
  - Pandora
  - Core LArSoft project

The experiments define requirements, schedules, priorities
What is LArSoft?

• A project / collaboration
  - Provide an integrated, art-based, experiment-agnostic set of software tools for LAr neutrino experiments to perform simulation, reconstruction and analysis
    • The core LArSoft (“project”) team maintains infrastructure, architecture, interfaces, coordination, code management and distribution
    • Experiment partners provide technical requirements, development effort and coordination, required experiment-specific plug-ins and configuration

• A body of code
  - Core LArSoft products
    • Experiment-agnostic data structures, algorithms, interfaces, etc
    • Lives in a set of repositories managed by the core LArSoft team
  - Experiment-specific components
    • Detector-specific geometry descriptions, electronics response functions, calibration functions, etc.
    • Live in repositories managed by the experiments

To understand the code and how it is made detector-agnostic, start with the operation of a LAr TPC
Operation of a single-phase LAr TPC
Operation of single-phase LAr TPC

LAr volume

Cathode plane

Electric field

Anode planes
Operation of single-phase LAr TPC

Ionized Ar produces scintillation light.

“Flash” arrives at photo detectors ~10s of ns (...the “early” light, at least)
Ionized Ar produces scintillation light.

"Flash" arrives at photo detectors ~10s of ns (...the "early" light, at least)
Ionized Ar produces scintillation light.

“Flash” arrives at photo detectors ~10s of ns (...the “early” light, at least)
Neutrino interacts with Ar nucleus

Charged secondaries ionize the Ar

Electrons drift in the electric field toward anode wires

\[ v_{\text{drift}} \approx 1 \text{ – few mm/\mu s} \]

Max drift time \(~ ms!!\)
Operation of single-phase LAr TPC

Neutrino interacts with Ar nucleus

Charged secondaries ionize the Ar

Electrons drift in the electric field toward anode wires

\[ v_{\text{drift}} \approx 1 - \text{few mm/}\mu\text{s} \]

Max drift time \( \sim \text{ms} \)!!
Operation of single-phase LAr TPC

Neutrino interacts with Ar nucleus

Charged secondaries ionize the Ar

Electrons drift in the electric field toward anode wires

\[ v_{\text{drift}} \approx 1 - \text{few mm/\mu s} \]

Max drift time ~ ms!!
Neutrino interacts with Ar nucleus
Charged secondaries ionize the Ar
Electrons drift in the electric field toward anode wires
\[ v_{\text{drift}} \approx 1 \text{ – few mm/\mu s} \]
Max drift time \( \sim \text{ms} \)!!
Operation of single-phase LAr TPC

- **LAr volume**
- **Cathode plane**
- **Electric field**
- Ionization electrons

Electrons pass through induction planes. Induce (small) bipolar signals.
Operation of single-phase LAr TPC

LAr volume

Cathode plane

Electric field

Electrons pass through induction planes.

Ionization electrons

Induce (small) bipolar signals
Operation of single-phase LAr TPC

(Large) unipolar pulses on collection plane wires
The job of the reconstruction:

To start with this...

wire 0
wire n

Cathode plane

Electric field
Operation of single-phase LAr TPC

The job of the reconstruction:

...and get to this:

And to get here...
Operation of single-phase LAr TPC

...you need to reconstruct this picture.
So now start from the raw signals, and walk through the general process, data structures needed to get here.
Primary reconstruction workflow and data structures
Reconstruction workflow and data structures

Start with the wire data.

LAr volume

Cathode plane

Track

Shower

Track

Shower
Reconstruction workflow and data structures

Start with the wire data.

(Large) unipolar pulses on collection plane wires
Reconstruction workflow and data structures

Start with the wire data.

Raw wire data: `raw::RawDigit`
All ACD values for a "channel ID"
Now need to look for signals associated with particles in the LAr

Two problems to solve first:
1) Induction plane signals are completely different from those on collection wires. (So use two algorithms??)
2) RawDigits are not calibrated

Raw wire data: raw::RawDigit
All ACD values for a “channel ID”
Wire calibration and deconvolution

• Calibration
  - Only pedestal subtraction at this phase. (Channel gains come later...)

• Deconvolution
  - The inverse of the following problem
    \[ y(t) = (h * x)(t) + n(t) \]
    - where \( y(t) \) = measured output signal (raw digits)
    - \( (h * x)(t) \) = convolution of impulse response \( h(t) \) and (unknown) input signal \( x(t) \)
    - \( n(t) \) = noise (unknown)
  - Can extract an optimal estimate of the signal given:
    - impulse response of the front-end electronics
    - estimated mean power spectrum for the signal and the noise (i.e., the signal-to-noise ratio)

Perform the calculation in the frequency domain

Performed by a number of classes:
- CalWire
- CalROI
- SignalShapingService
- SignalShaping
- LArFFT

(Experiment-specific)
Deconvolved signals in recob::Wire are unipolar, so can be treated with the same algorithm for all wire planes.

Deconvolved wire data: recob::Wire + art::Assns<RawDigit, wire>

All corrected / deconvolved ACD values for a “channel ID”
Reconstruction workflow and data structures

Deconvolved signals in recob::Wire are unipolar, so can be treated with the same algorithm for all wire planes.

For “ROI” algorithms, keep only the values in “regions of interest” (i.e., zero-suppressed).

Deconvolved wire data: recob::Wire + art::Assns<RawDigit, wire>

All corrected / deconvolved ACD values for a “channel ID”
The next step is to group together the ADC values on each wire that correspond to the ionization associated with a single particle (...more or less...) as it traverses the measurement volume for that wire. This is “hit-finding”. 
The next step is to group together the ADC values on each wire that correspond to the ionization associated with a single particle (...more or less...) as it traverses the measurement volume for that wire. This is “hit-finding”.

The output of hit-finding: `recob::Hit + Assns<Wire,Hit>, Assns<RawDigit,Hit>`

All ADC values on a given wire attributed to a single particle, and the arrival time of ionization relative to a common (arbitrary) $t_0$.

Hits are used as input to estimate:
1) the actual position of the hit
2) the actual charge in the hit, and therefore the energy deposition within the measurement volume of the wire.
The next step is to group together the ADC values on each wire that correspond to the ionization associated with a single particle (...more or less...) as it traverses the measurement volume for that wire.

This is “hit-finding”.

The output of hit-finding: recob::Hit + Assns<Wire,Hit>, Assns<RawDigit,Hit>

All ADC values on a given wire attributed to a single particle, and the arrival time of ionization relative to a common (arbitrary) t0

Hit-finding performed by:
CCHitFinder
GausHitFinder
RFFHitFinder
...
Now need to start grouping hits into “clusters” that represent the ionization of a single physical entity.

This “clustering” procedure is (usually) performed in 2D.
Now need to start grouping hits into “clusters” that represent the ionization of a single physical entity.

This “clustering” procedure is (usually) performed in 2D.

The output of 2D clustering: recob::Cluster + Assns<Hit,Cluster>
All hits attributed to a single physical entity, (relative) start/end position, some shape parameters
Now need to start grouping hits into “clusters” that represent the ionization of a single physical entity.

This “clustering” procedure is (usually) performed in 2D.

Algorithms optimized for track-like clusters.

2D cluster-finding performed by:
- ClusterCrawlerAlg
- DBScanAlg
- fuzzyClusterAlg
  ...

The output of 2D clustering: \( \text{recob::Cluster + Assns<Hit,Cluster>} \)
All hits attributed to a single physical entity, (relative) start/end position, some shape parameters
Now need to start grouping hits into “clusters” that represent the ionization of a single physical entity. This “clustering” procedure is (usually) performed in 2D.

There is also at least one 3D clustering algorithm, Cluster3D.

It produces associated 2D recob::Cluster objects among other things.

The output of 2D clustering: recob::Cluster + Assns<Hit,Cluster>
All hits attributed to a single physical entity, (relative) start/end position, some shape parameters.
Reconstruction workflow and data structures

Given the 2D clusters, can now combine them into 3D objects.

“Track-finding” is the first part of this process.

The output of tracking: recob::Track + Assns<Cluster,Track>
Estimated points + direction + covariance along trajectory
Tracks can also have associated recob::Vertex objects and recob::PFParticle objects (preliminary particle flow hypotheses)
Given the 2D clusters, can now combine them into 3D objects.

“Track-finding” is the first part of this process.

The output of tracking:  
recob::Track + Assns<Cluster,Track>  
Estimated points + direction + covariance along trajectory  
Tracks can also have associated recob::Vertex objects and recob::PFPParticle objects (preliminary particle flow hypotheses)
Clusters can also be part of showers. Finding shower-like clusters is sometimes done at the same time as the shower-finding itself.

Either way, this step is “shower-finding”

This is also usually the start of calorimetric measurements

Shower-finding output:  
- recob::Shower + anab::Calorimetry  
- Assns<Cluster,Shower> + Assns<Hit,Shower>  
  Shower parameters, energy estimates
Clusters can also be part of showers. Finding shower-like clusters is sometimes done at the same time as the shower-finding itself.

Either way, this step is “shower-finding”

This is also usually the start of calorimetric measurements

**Shower-finding algorithms:**
- ShowerFinder
- ShowerReco
- ShowerReco3D
  ...

**Shower-finding output:**
- `recob::Shower` + `anab::Calorimetry`  
  Assns<Cluster,Shower> + Assns<Hit,Shower>

Shower parameters, energy estimates
Without the interaction $t_0$, the absolute position of energy deposition cannot be determined.

Use scintillation light to determine the interaction time in a procedure called “flash-finding”
Without the interaction $t_0$, the absolute position of energy deposition cannot be determined. Use scintillation light to determine the interaction time in a procedure called “flash-finding”.

Start with optical raw data

Optical raw data: raw::OpDetWaveform
Optical reconstruction starts with “hit-finding” (analogous to wire hit-finding)

Without the interaction $t_0$, the absolute position of energy deposition cannot be determined.

Use scintillation light to determine the interaction time in a procedure called “flash-finding”

Flash-finding output: recob::OpHit + recob::OpFlash
Pulse heights, arrival times, event $t_0$, flash centroid
Optical reconstruction starts with “hit-finding” (analogous to wire hit-finding), followed by flash-finding, which estimates interaction $t_0$.

Without the interaction $t_0$, the absolute position of energy deposition cannot be determined.

Use scintillation light to determine the interaction time in a procedure called “flash-finding”.

Flash-finding output: `recob::OpHit + recob::OpFlash`
- Pulse heights
- Arrival times
- Event $t_0$
- Flash centroid
Without the interaction $t_0$, the absolute position of energy deposition cannot be determined.

Use scintillation light to determine the interaction time in a procedure called “flash-finding”.

Optical reconstruction starts with “hit-finding” (analogous to wire hit-finding), followed by flash-finding, which estimates interaction $t_0$.

Flash-finding algorithms: OpFlashAlg

Flash-finding output: recob::OpHit + recob::OpFlash
Pulse heights, arrival times, event $t0$, flash centroid
Secondary reconstruction
“Analysis-phase” reconstruction

- Cosmic ray removal
  - Particularly important for surface detectors
    - SBN detectors at Fermilab
    - Test beam detectors
  - Employs track-finding, clustering, flash-track and flash-cluster matching
  - Representative algorithms: CosmicTrackTagger, BeamFlashTrackMatchTagger...
    - Output: anab::CosmicTag

- Calorimetric measurements
  - Energy and dE/dx estimates for Tracks
  - Representative algorithms: CalorimetryAlg, TrackCalorimetryAlg
    - Output: anab::Calorimetry
“Analysis-phase” reconstruction

- Momentum estimation and particle identification
  - Use range, dE/dx and multiple Coulomb scattering of tracks
  - Representative algorithms: Chi2PIDAlg, PIDAAlg
  - Output: anab::ParticleID, Assns<Track, ParticlePID>, or TTree
Other complications

• Space-charge distortions
  - Ion drift mobilities are about $10^6 \times$ smaller than that of electrons
    • Cation drift velocities are $\sim$nm / $\mu$s !
  - High cosmic ray rate for surface detectors introduces significant positive ion load
    • At MicroBooNE, field distortions could reach cm scales
  - Need to map and correct for these
  - A common service exists to access the offsets

• Charge attenuation
  - Electron lifetime can be comparable to maximum drift time
  - Effective gain will be drift-length dependent
  - Expect to see significant reduction in SNR with drift distance
Other complications

- Hit disambiguation
  - DUNE TPCs have wrapped induction wires
    - See signals in two TPCs, and in many cases, multiple places within the same TPC
  - Some LArSoft nomenclature:
    - “Wire”: a segment of a physical wire in an anode plane
      - A geometric concept
    - “Channel”: a readout channel connected to one or more “wires”
      - A DAQ concept
Other complications

• Hit disambiguation
  - DUNE TPCs have wrapped induction wires
    • See signals in two TPCs, and in many cases, multiple places within the same TPC
  - Some LArSoft nomenclature:
    • “Wire”: a segment of a physical wire in an anode plane
      - A geometric concept
    • “Channel”: a readout channel connected to one or more “wires”
      - A DAQ concept

One channel (same physical wire)
Other complications

- Hit disambiguation
  - DUNE TPCs have wrapped induction wires
    - See signals in two TPCs, and in many cases, multiple places within the same TPC

- Some LArSoft nomenclature:
  - “Wire”: a segment of a physical wire in an anode plane
    - A geometric concept
  - “Channel”: a readout channel connected to one or more “wires”
    - A DAQ concept
Other complications

- **Hit disambiguation**
  - DUNE TPCs have wrapped induction wires
    - See signals in two TPCs, and in many cases, multiple places within the same TPC
  - Introduce a disambiguation step to deal with this
    - Resolves the TPC ambiguity of each induction hit
    - Currently performed after hit-finding
      - Existing algorithms use timing information and neighboring activity
Other complications

- Dual-phase LAr TPCs
  - Under development / consideration by DUNE
  - Do not yet understand potential implications for LArSoft
Detector simulation in LArSoft
Simulation workflow

Three phases, typically run separately

Beam simulation
Momentum, spatial, angular distribution of neutrinos from target incident on detector
Output is a “flux file”.

Legend

= processes
= data

From W Seligman
Simulation workflow

Event generation
- Produces final state secondaries of selected physics processes for incident neutrinos.
- Output is `simb::MCTruth`

Lots of options available for the event generator!!
- Can run different generators using the same flux files as input.

Legend
- = processes
- = data

From W Seligman
Simulation workflow

Three phases, typically run separately

“Full” Detector simulation
Output is fully simulated raw data + MC truth information about what was simulated.

Legend
- = processes
- = data

From W Seligman
Simulation workflow

The full detector simulation includes two separable sub-phases

Detector response
Field effects, noise, transfer function, etc.
Output is fully simulated wire and PMT data.

Detector simulation:
Currently GEANT4 (LArG4 module)
Particle propagation/interactions, charge/photon transport.
Output is MC truth information in simb::MCParticle, sim::SimPhoton, sim::SimChannel, sim::AuxDetSimChannel,
Simulation workflow

Most detect-specific customizations go into the detector response

Detector response
Field effects, noise, transfer function, etc.
Output is fully simulated wire and PMT data.

Detector simulation:
Currently GEANT4 (LArG4 module)
Particle propagation/interactions, charge/photon transport.
Output is MC truth information in simb::MCParticle, sim::SimPhoton, sim::SimChannel, sim::AuxDetSimChannel,
LArSoft design principles
LArSoft design principles and practices

- Detector interoperability
  - The most important design objective for the LArSoft project
  - Define (and use!!) common interfaces for accessing detector-specific configuration information and functionality
  - Applies to geometry, channel mappings, LAr properties, E-field map...

- Separation of framework and algorithm code
  - Encapsulate algorithms, configuration, tools and utilities into a layer that is independent of the art framework
  - Will describe reasons and techniques to achieve this later in the course

General disclaimer:
In examining the code, you may note that only a portion currently adheres to these principles.
- An on-going architecture review project is intended to address this

Strongly encourage people to adopt these practices for all new code
LArSoft design principles and practices

- **Use of standardized algorithm interfaces**
  - Define standard interfaces for well-defined steps in the workflow to promote modularity, layering of algorithms

- **Modularity**
  - Build sophistication by applying algorithms in a layered, iterative structure

- **Design / write testable units of code**
  - Include unit and integration testing in the development process
  - Follow the practice of continuous integration
    - Perform automated, broad-scale testing at frequent intervals in order to catch unintended side-effects quickly
LArSoft design principles and practices

• Document code in the source files
  - See many files with no comments at all
  - At very least, need the purpose of the file, how it is used, pre-requisites, assumptions,
Using LArSoft
Supported platforms

- **Scientific Linux**
  - SLF6
  - Have also installed / run this code under:
    - SLC6 (CERN)
    - Redhat 6 (SLAC)

- **Mac OSX**
  - Mavericks and Yosemite
    - ups qualifiers d13:noifdh and d14:noifdh respectively

- **Installation instructions**
  - See links in release notes available at
Releases

- Two types of releases
  - Integration
    - Created weekly or on demand for special purposes
    - Contents approved at Coordination Meetings
      - Head of develop + additional branches approved at a CM or via email
    - May be removed without notice after about a month
      - In practice, we announce our intentions in advance
  - Production
    - Any release designated as “production” by an experiment
    - Created on demand (but usually on the weekly schedule)
    - Contents approved by the experiment declaring production
      - Typically also coordinated through the CM to keep other experiments informed
    - Production releases are retained on disk indefinitely

- List of all available tagged releases
How to set up and run art/LArSoft

• First point to note
  - LArSoft is designed to be run by experiments
    • Need detector-specific parts to run it
    • So start with the code of your experiment
      - MicroBooNE uboonecode
      - DUNE lbnecode (changing name soon!!)
      - SBND lar1ndcode (changing??)
      - LArIAT lariatsoft

  The setup procedures for each are different
  so refer to the setup relevant setup instructions

• Second point to note
  - Do not need a “working area” to run LArSoft. Just need to set up the
    appropriate products + a fcl file
How to set up and run art/LArSoft

• The most simple scenario: run from a tagged LArSoft release
  - First, set up the working environment:
    ```
    # Set up ups
    source <ups location>/setup
    
    # Set up the working environment
    # for your experiment
    <the setup procedure for your experiment here>
    
    # Set up a LArSoft release
    setup larsoft v04_16_00 -q e7:<prof|debug>
    ```
  
  • In most cases, experiment-specific setup scripts will include all of the above
  • Note that setting up for development requires additional steps
  - Now run art (the LArSoft version is called lar)
    ```
    # List art command-line options
    lar --help
    
    # Now run LArSoft
    
    lar -c <some fcl file>.fcl [-i <input file>]
    ```

That's it! ...if you have a fcl file and have no code to change
Setting up and running LArSoft as a developer

- This is the second case of running LArSoft,
  - Applies to people developing LArSoft, or writing analysis software using art
  - Will defer this to the next session, when Saba will talk about how to contribute to LArSoft and use the build tools
Getting LArSoft to work with a new detector

• LArSoft is really a toolkit
  - It requires a certain amount of detector-specific information and plug-in functionality in order to work

• The minimum needed to run LArSoft for a new detector
  - Define the geometry for the new detector in a GDML file
  - Customize E-field, drift velocity, readout parameters, etc, as needed
  - Customize digitization for simulation, as needed
  - Write a fcl file
  - lar -c sim_new_det.fcl → simulated data; lar -c <reco...>.fcl → results!!

...A bit over-simplified, but this is basically what happens
Resources
LArSoft Redmine site

https://cdcvs.fnal.gov/redmine/projects/larsoft

Redmine sites are called “projects”

This is the home page for the LArSoft Redmine project

Tabs across the top link to different types of content

Aug. 7, 2015

Introduction to LArSoft

E. Snider

74
The LArSoft Redmine site

Redmine sites are called “projects”

https://cdcvvs.fnal.gov/redmine/projects/larsoft

This is the home page for the LArSoft Redmine project

Tabs across the top link to different types of content

The most useful tabs:
- Wiki
- Issues
- New Issues
- Repository

Aug. 7, 2015
Introduction to LArSoft
E. Snider
LArSoft wiki

https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki

General information and documentation

Instructions, list of releases, and other useful information.

Quick-start guide to using and developing LArSoft code
LArSoft issue tracker

https://cdcvs.fnal.gov/redmine/projects/larsoft/issues

We track bugs, problems with LArSoft-related infrastructure, requests for support and new features, questions...

Open a new ticket using the “New Issue” tab if you have any of the above.

Create a new issue using this tab.

Must be logged into Redmine using your Fermilab services account and password.
LArSoft Redmine code browser

https://cdcvs.fnal.gov/redmine/projects/larreco/repository

Each LArSoft repository lives in a separate Redmine project which is specified here.
Navigating between LArSoft sub-projects

https://cdcvs.fnal.gov/redmine/projects/larsoft/repository

Can use the project navigation pull-down to get to the desired project.
Navigating between LArSoft sub-projects

Can use the project navigation pull-down to get to the desired project.
LArSoft Indico site

- Slides and documents from meetings are posted to Indico
  
  https://indico.fnal.gov/categoryDisplay.py?categId=233

  - Or from the Indico home page:  https://indico.fnal.gov/index.py, follow “Experiments”, then “LArSoft” links to arrive at the LArSoft page

Can upload slides on the page for the particular meeting...

...but, must be logged in using your Indico account and password
Resources

- LArSoft dOxygen documentation system:

- LAr reconstruction software forum: http://www.larforum.org/forum
  - Help with general problems in LAr software
  - So far almost entirely focused on LArSoft...

- LArSoft email list: larsoft@fnal.gov
  - General announcements. Often technical questions also.

- LArSoft Coordination Meeting
  - Bi-weekly at 13:00 Central Time. July 28 is the next one.
  - Remote connections via ReadyTalk. Slides posted to LArSoft Indico site.

- LArSoft wiki: https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki
  - Quick-start guide to using and developing LArSoft code
  - See also https://cdcvs.fnal.gov/redmine/projects/uboonecode/wiki

- LArSoft issue tracker
Core LArSoft support team

- **Core team members**
  - Technical lead: Erica Snider  
    erica@fnal.gov
  - Project manager: Ruth Pordes  
    ruth@fnal.gov
  - Lead developer: Gianluca Petrillo  
    petrillo@fnal.gov
  - Developer: Saba Sehrish  
    ssehrish@fnal.gov
  - Code management and distribution: Lynn Garren  
    garren@fnal.gov
  - CI operations and testing support: Vito di Benedetto  
    vito@fnal.gov

Email / visit any of the project team!!
The end
Backup
Event generators

- **Flux**
  - GENIE: Standard for neutrino interactions
  - CRY: Cosmic rays
  - NuWro
  - TextFileGen: HepMC text file input
  - GIBUU
  - LightSource: Optical photons
  - NUANCE: For comparison with MiniBooNE
  - Single: Particle "gun"

- **MCTruth**: Detector interactions

From W Seligman
Detector simulation

From W Seligman
Simulation task workflow

Detector response and digitization

- Waveform generation
- Gains and pedestals
- Convert to ADC counts

SimWire applies corrections for:
- Electric field
- Electronics shaping
LArSoft design principles and objectives

- Detector interoperability
  - The most important design objective for the LArSoft project
  - Requires care to define (and use!!) common interfaces for accessing detector-specific configuration information
  - Good example: access to detector geometry information
    - A single interface that accommodates different (albeit very similar) geometries
      - Most differences a matter of configuration only
      - Also have detector-specific implementations of the interface where needed
    - Carefully avoid implied geometrical assumptions in algorithms
      - Position of the first plane or wire, the wire spacing, etc.
    - Introduced structures to facilitate generic loops over geometrical elements
      - Define detector / DAQ element IDs at all levels of detector geometry hierarchy
      - Can thereby avoid explicit reference to indices for loops, etc.
  - Also applies to
    - Access to calibration data, LAr properties, detector properties, E-field map, handling of common metadata for data files...
  - Have compiled a long list of do's and don'ts to ensure interoperability
LArSoft design principles and objectives

• Separation of framework and algorithm code
  - Encapsulate algorithms, configuration, tools and utilities into a layer that is independent of the framework
  - Why??
    • Allow testing of small units of algorithm code outside the framework
    • Provide greater flexibility in using algorithms
    • To provide a means of integrating LArSoft code (data products and algorithms) with external frameworks
      - e.g., LArLite used by MicroBooNE for algorithm development, testing

See art guidance for writing modules for further discussion
LArSoft design principles and objectives

- Separation of framework and algorithm code
  - Encapsulate algorithms, configuration, tools and utilities into a layer that is independent of the framework
  - Why??
    - Allow testing of small units of algorithm code outside the framework
    - Provide greater flexibility in using algorithms
    - To provide a means of integrating LArSoft code (data products and algorithms) with external frameworks
      - e.g., LArLite used by MicroBooNE for algorithm development, testing
  
  See art guidance for writing modules for further discussion

All this code lives in LArSoft repositories
LArSoft design principles and objectives

• Separation of framework and algorithm code (cont’d)
  
  - How??
    
    • Adhere to particular design patterns for the “LArSoft / art interface” code
      - i.e., art modules and services (to be discussed later)
    
    • Access framework functionality only within the interface code
      - Includes finding input data, writing output data, retrieving any required services, making filter decision calls, etc.
    
    • Pass all required data, utility classes into algorithms, and all output data back out

General disclaimer: In examining the code, you may note that only a portion of the existing code adheres to this and the other design principles.

• The on-going architecture review project is intended to address this

Strongly encouraging people to adopt this practice for new code.
LArSoft design principles and objectives

• Standardized algorithm interfaces
  - Define standard interfaces for well-defined steps in the workflow so that:
    • Multiple algorithms that address specific problems can share interfaces
  - Promote greater modularity, layering of algorithms
  - Generally enhances flexibility of the code

• Modularity
  - Build sophistication by applying small, targeted algorithms in a layered, iterative structure
  - Leads to code that is more easily tested, more maintainable, more flexible
LArSoft design principles and objectives

- Continuous integration
  - A development scheme in which changes to the main branch of development are integrated and tested frequently
    - At every push to the develop branch
    - Every night
    - At every release
  - Primary LArSoft goals
    - Ensure that code performs as intended
    - Facilitate early detection of problems created in one experiment due to changes introduced by another experiment
    - Ensure that all major features in the develop branch work at all times
  - Are now operating a continuous integration system for LArSoft
    - Currently runs at every push to develop branch
    - Can be triggered manually to run on a non-develop branch of a user's choosing
See https://cdcvs.fnal.gov/redmine/projects/lar-ci/wiki for details