Introduction to LArSoft

Erica Snider, *Fermilab*
on behalf of SciSoft Team

LArSoft 2021 Reference, July 8

Outline

- Underlying principle of LArSoft
- The LArSoft Collaboration
- Operation of a single-phase LAr TPC
- Simulation and reconstruction in LArSoft
- Design principles and coding practices
- LArSoft physical design
- Code releases and distribution
- End-user / developer resources

Underlying principle of LArSoft

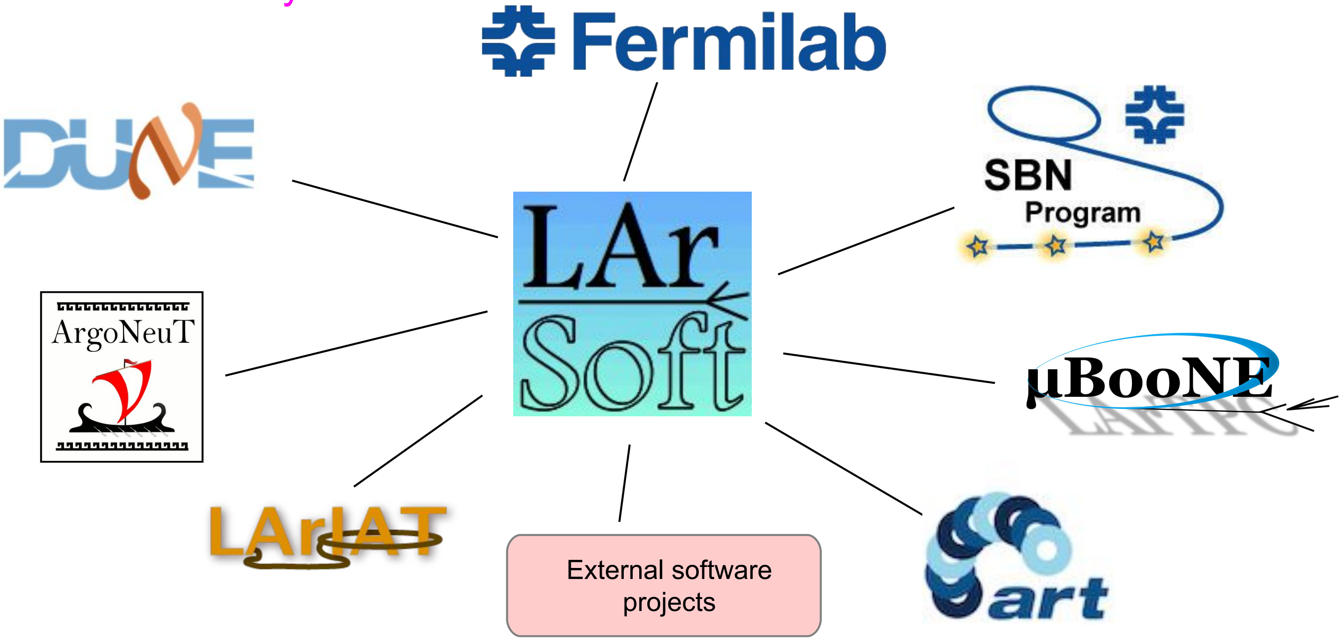
Exploit the similarity in the geometry and readout schemes that are common to many LArTPCs to create a set of infrastructure and algorithms for the simulation and reconstruction of LArTPC data that can shared across detectors

- Use common data structures and interfaces
- Express detector-specific differences via configuration
- Write algorithms that work for any / many LArTPCs

As a result, dramatically reduce the cost of developing this software for experiments that use LArTPC technology

The LArSoft Collaboration

Experiments, laboratories, software projects collaborating to produce, shared experiment-independent software for LArTPC simulation, reconstruction and analysis

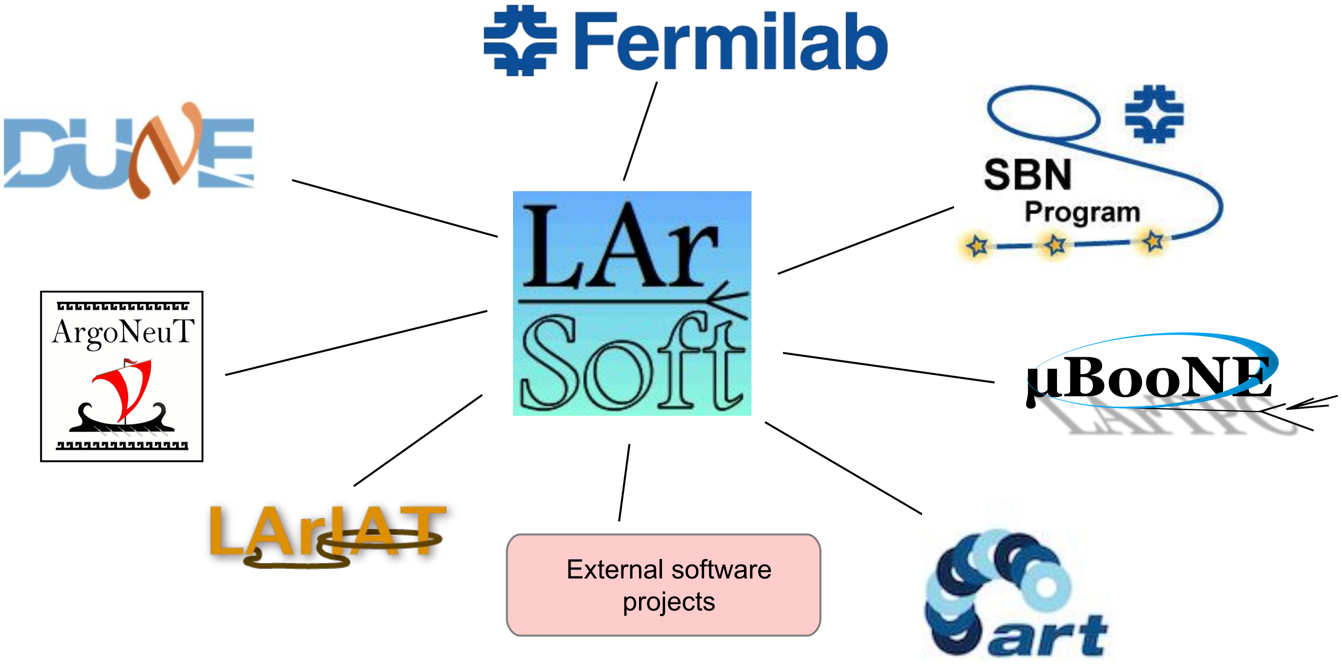


External software projects

The LArSoft Collaboration

The LArSoft “project”: a Fermilab-based group that

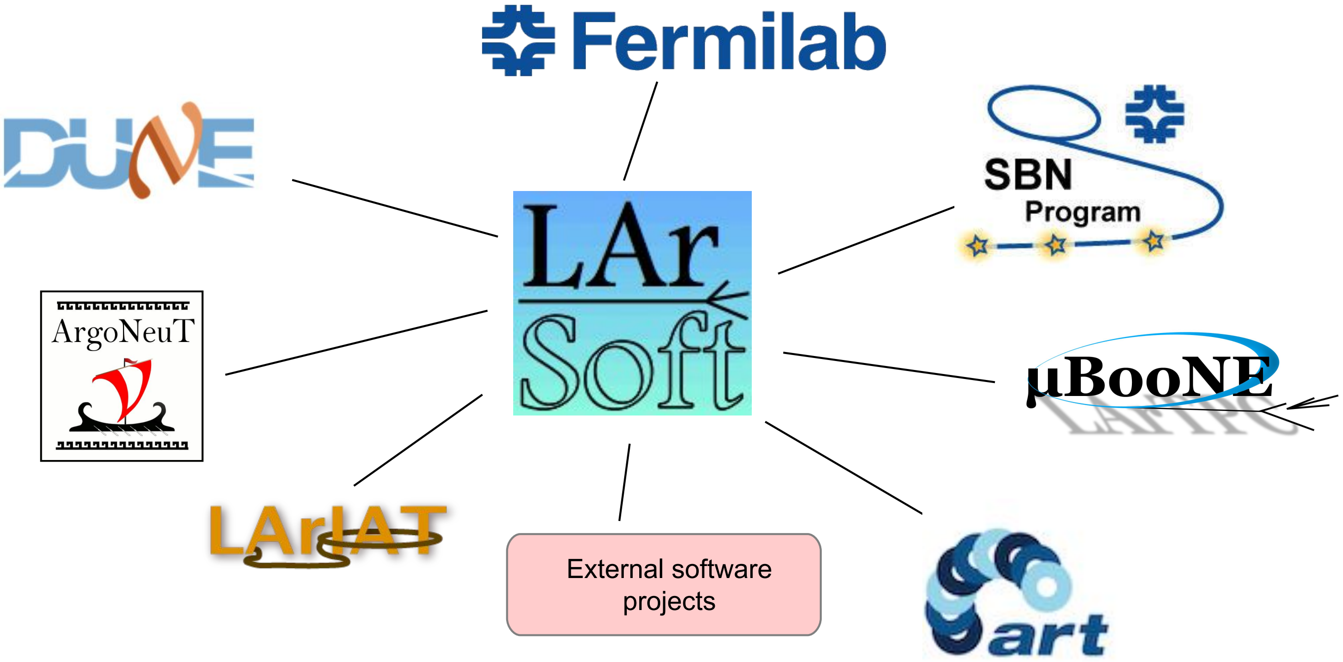
- maintains / develops the architecture
- provides user support, software expertise, release management



External software projects

The LArSoft Collaboration

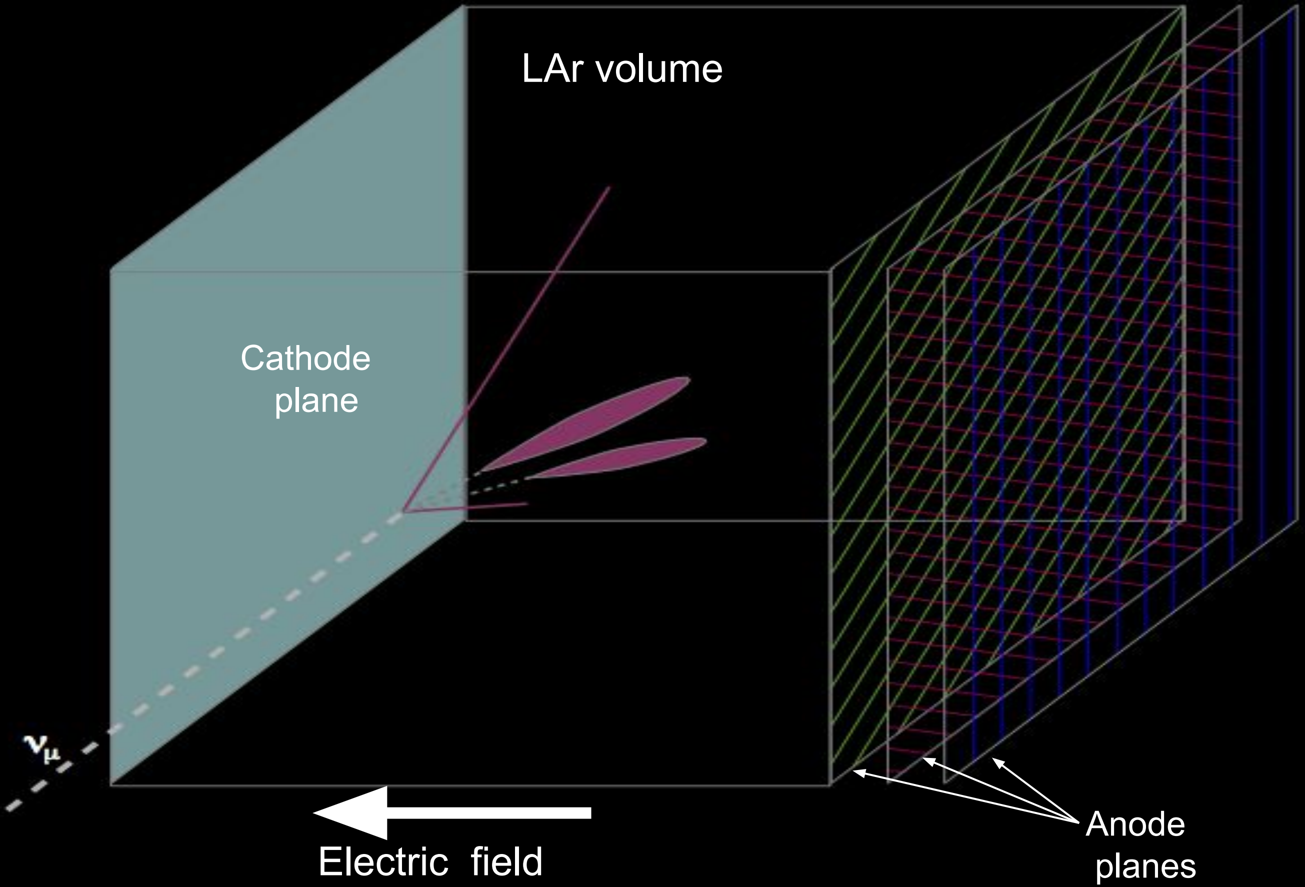
The body of shared software is also referred to as “LArSoft”



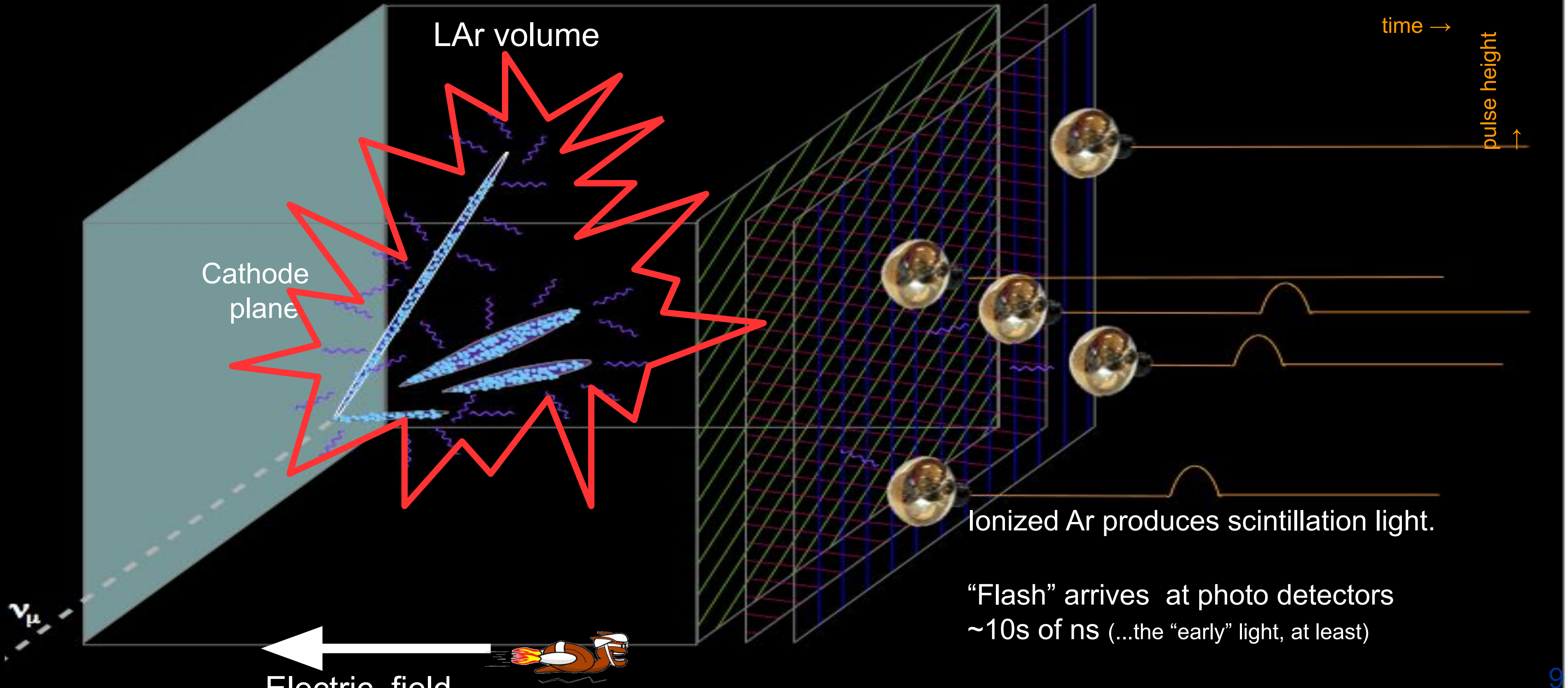
External software projects

Operation of a single-phase LAr TPC

Operation of single-phase LAr TPC



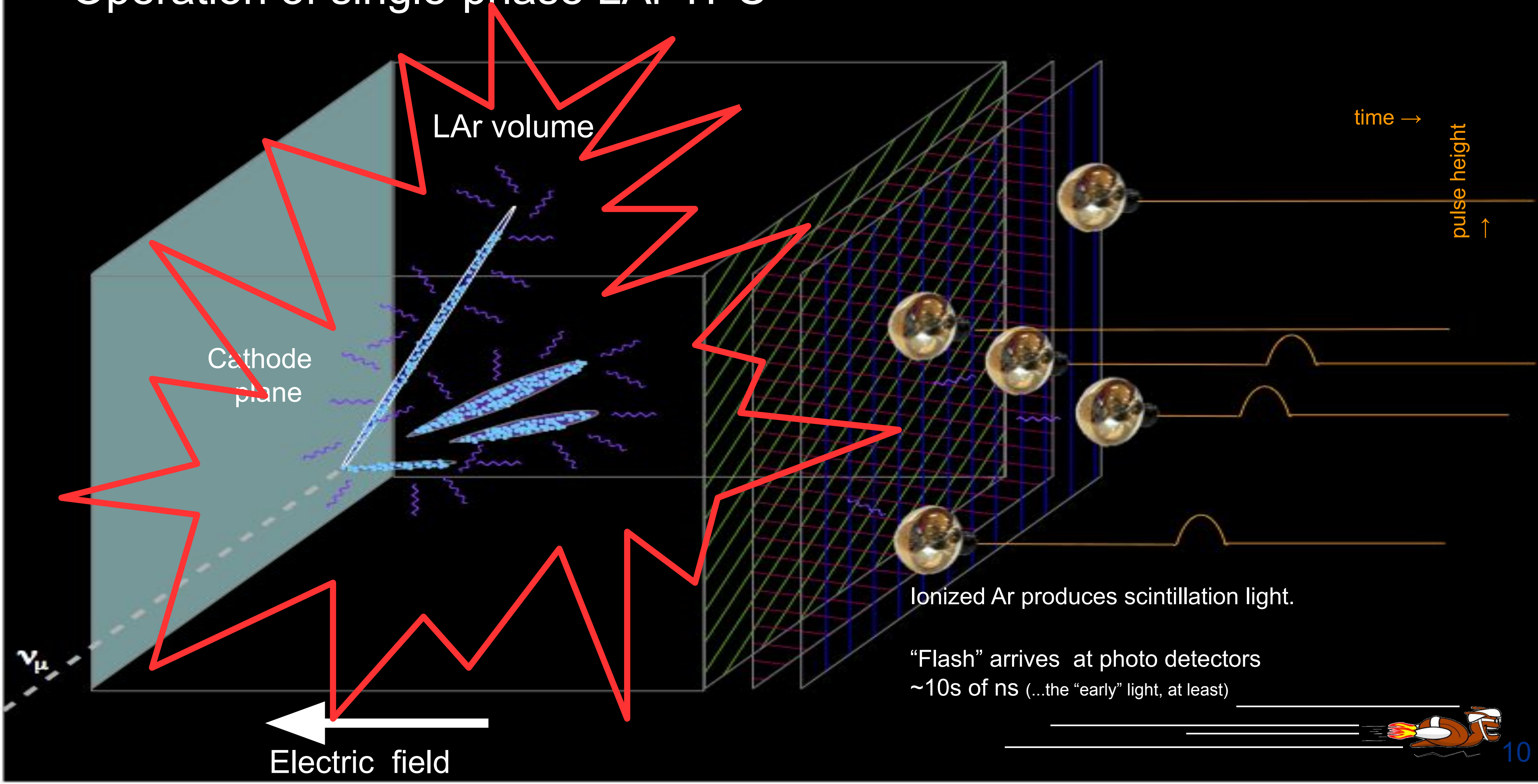
Operation of single-phase LAr TPC



Ionized Ar produces scintillation light.

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

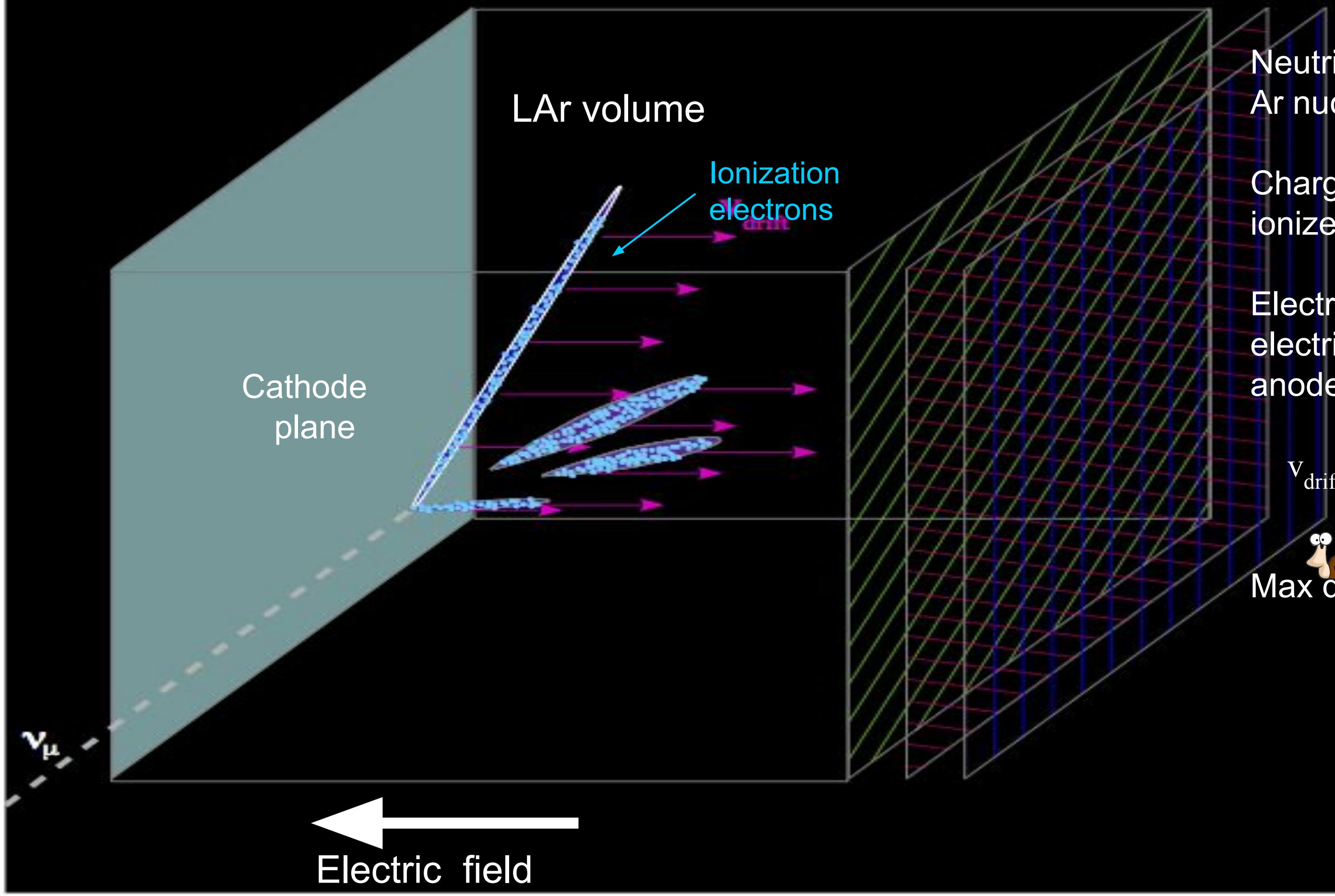
Operation of single-phase LAr TPC



Ionized Ar produces scintillation light.

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

Operation of single-phase LAr TPC



Neutrino interacts with Ar nucleus

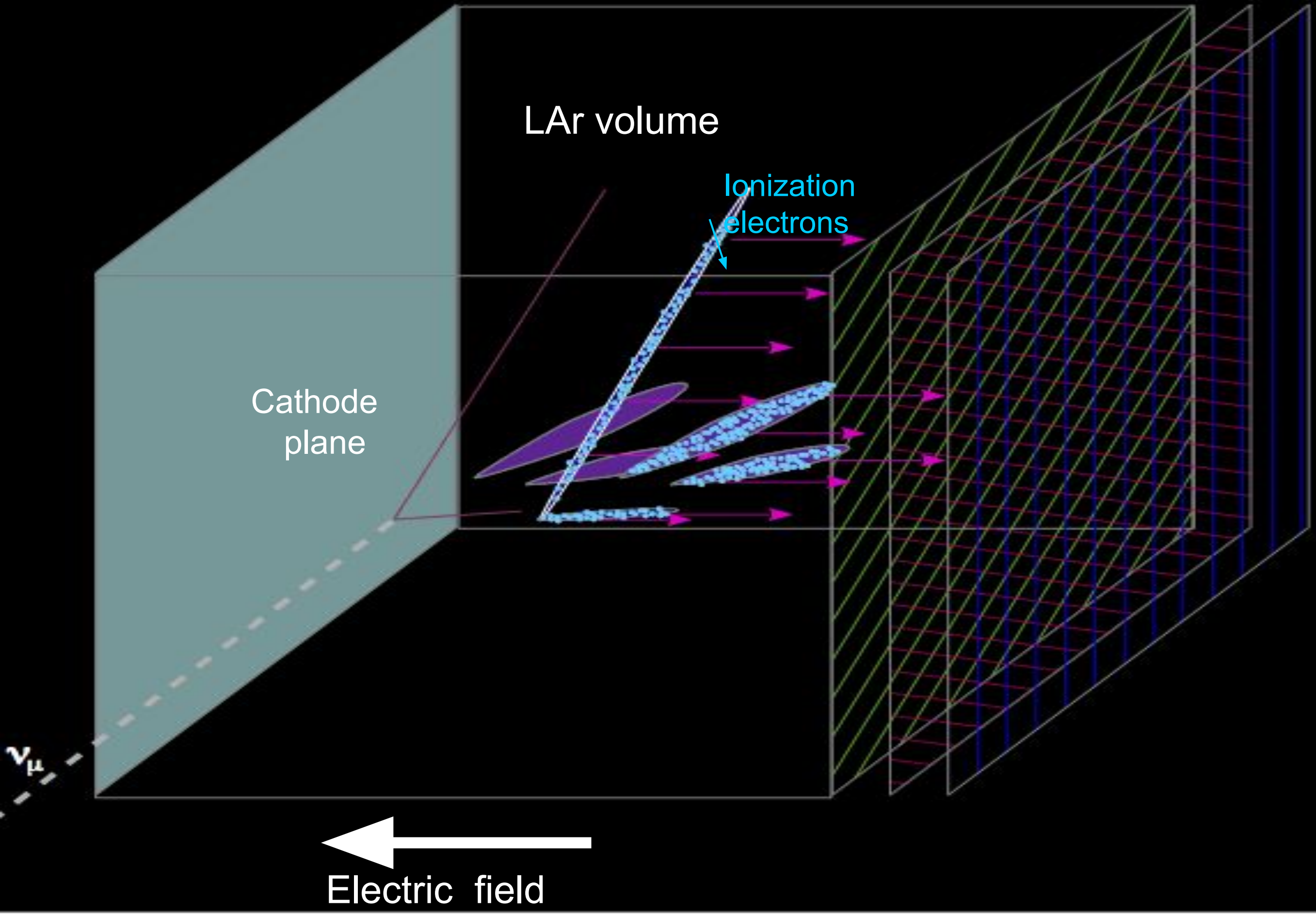
Charged secondaries ionize the Ar

Electrons drift in the electric field toward anode wires

$v_{drift} \approx 1$ – few mm/μs

Max drift time ~ ms!!

Operation of single-phase LAr TPC



Neutrino interacts with Ar nucleus

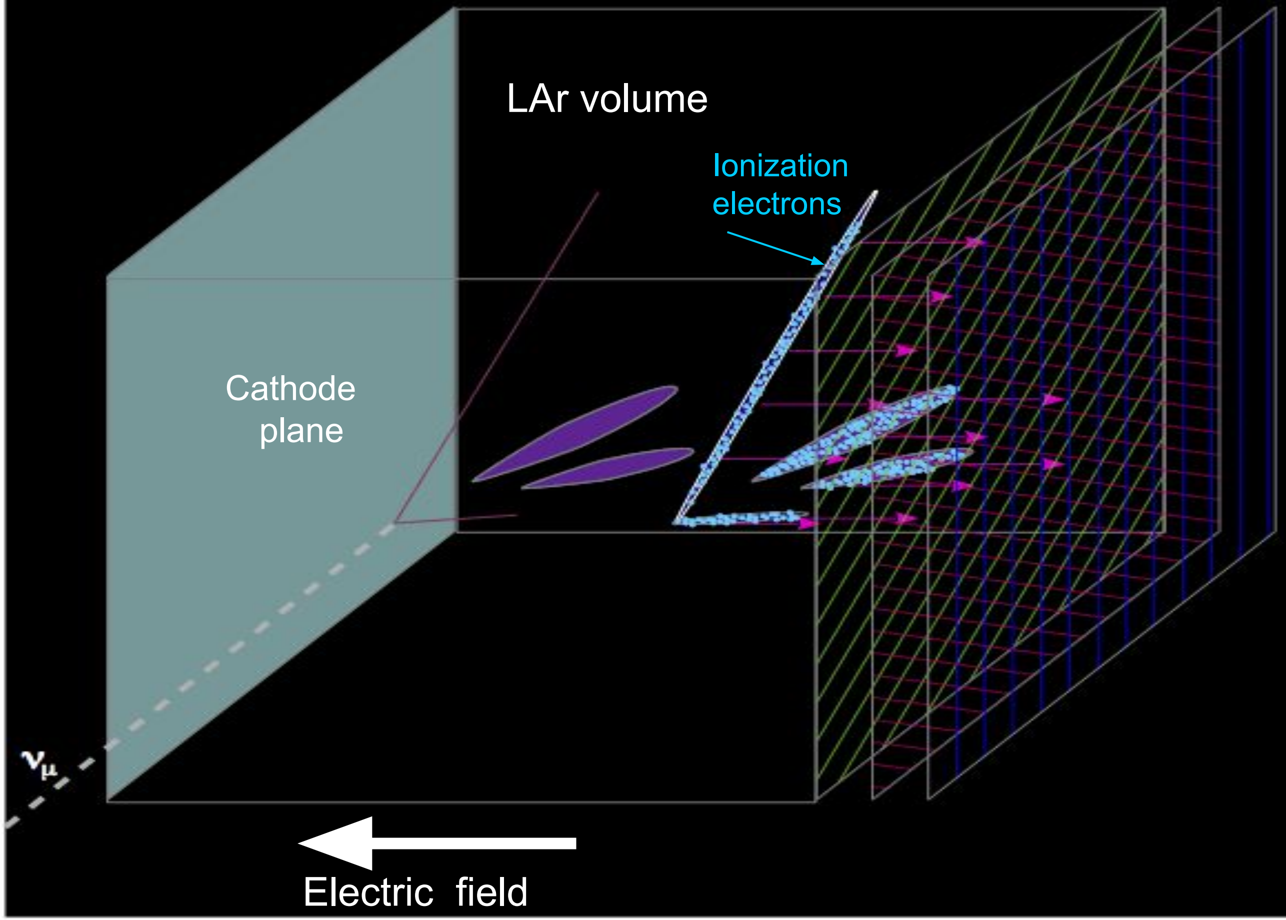
Charged secondaries ionize the Ar

Electrons drift in the electric field toward anode wires

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

Max drift time ~ ms!!

Operation of single-phase LAr TPC



Neutrino interacts with Ar nucleus

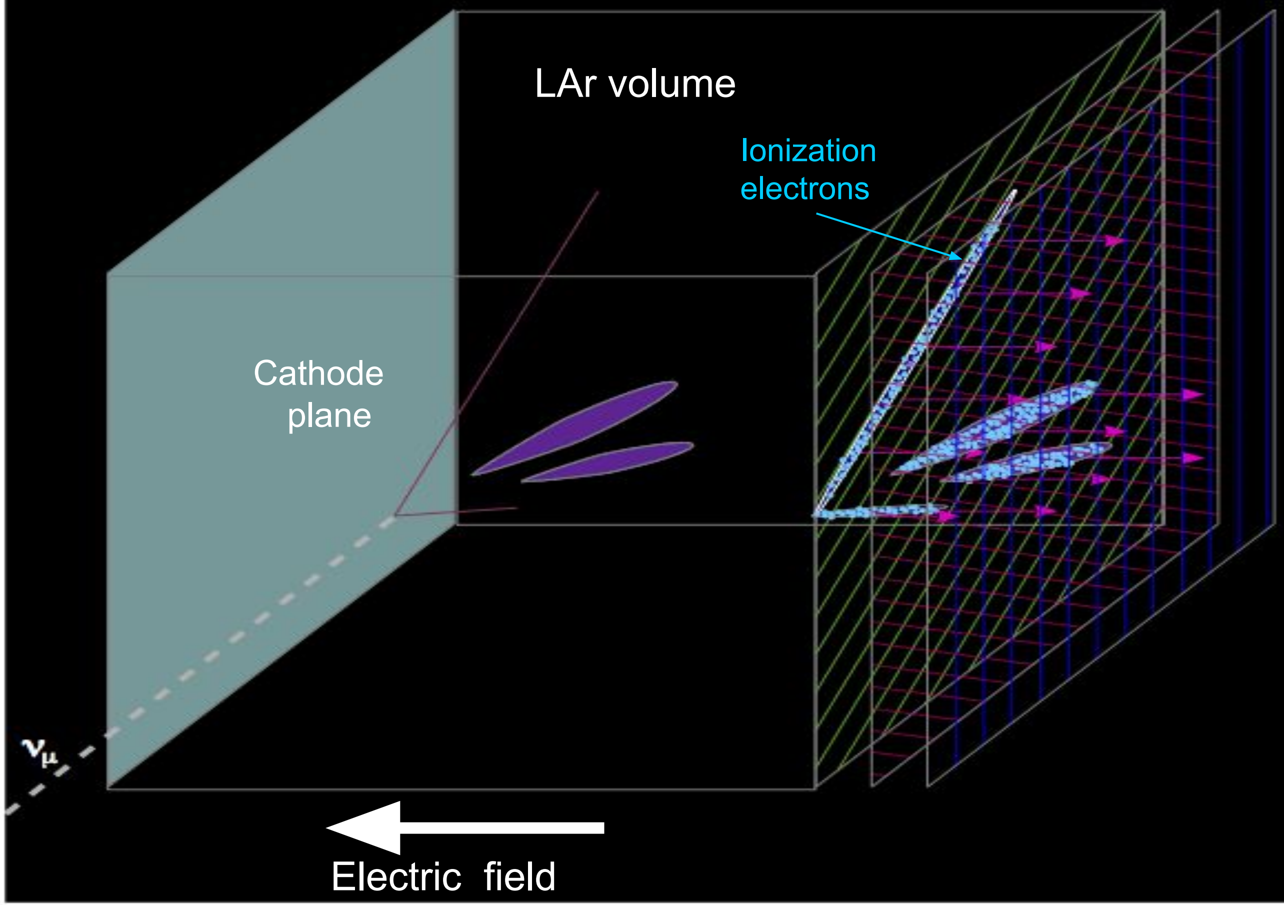
Charged secondaries ionize the Ar

Electrons drift in the electric field toward anode wires

v_{drift} ≈ 1 – few mm/μs

Max drift time ~ ms!!

Operation of single-phase LAr TPC



Neutrino interacts with Ar nucleus

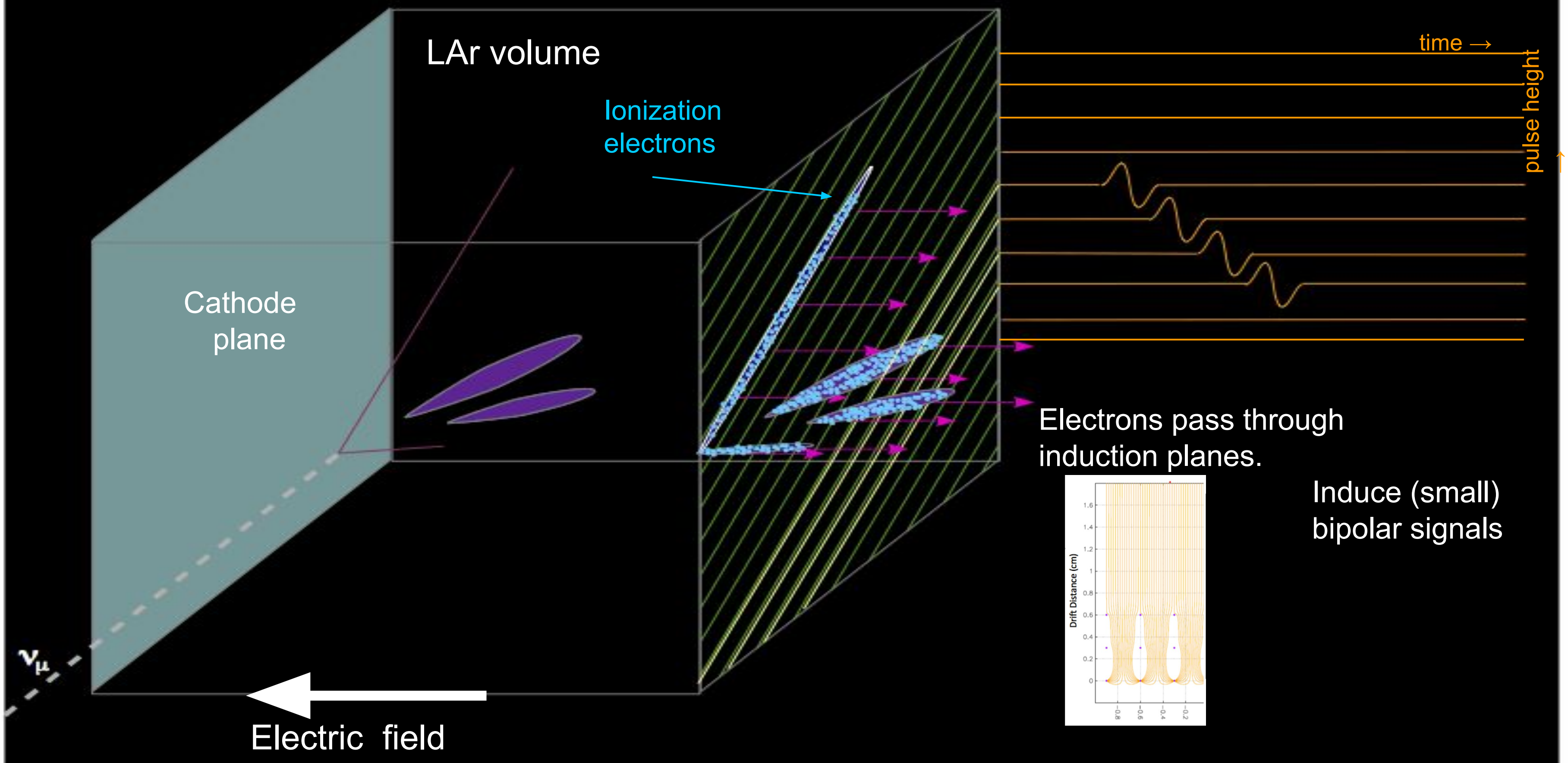
Charged secondaries ionize the Ar

Electrons drift in the electric field toward anode wires

v_{drift} ≈ 1 – few mm/μs

Max drift time ~ ms!!

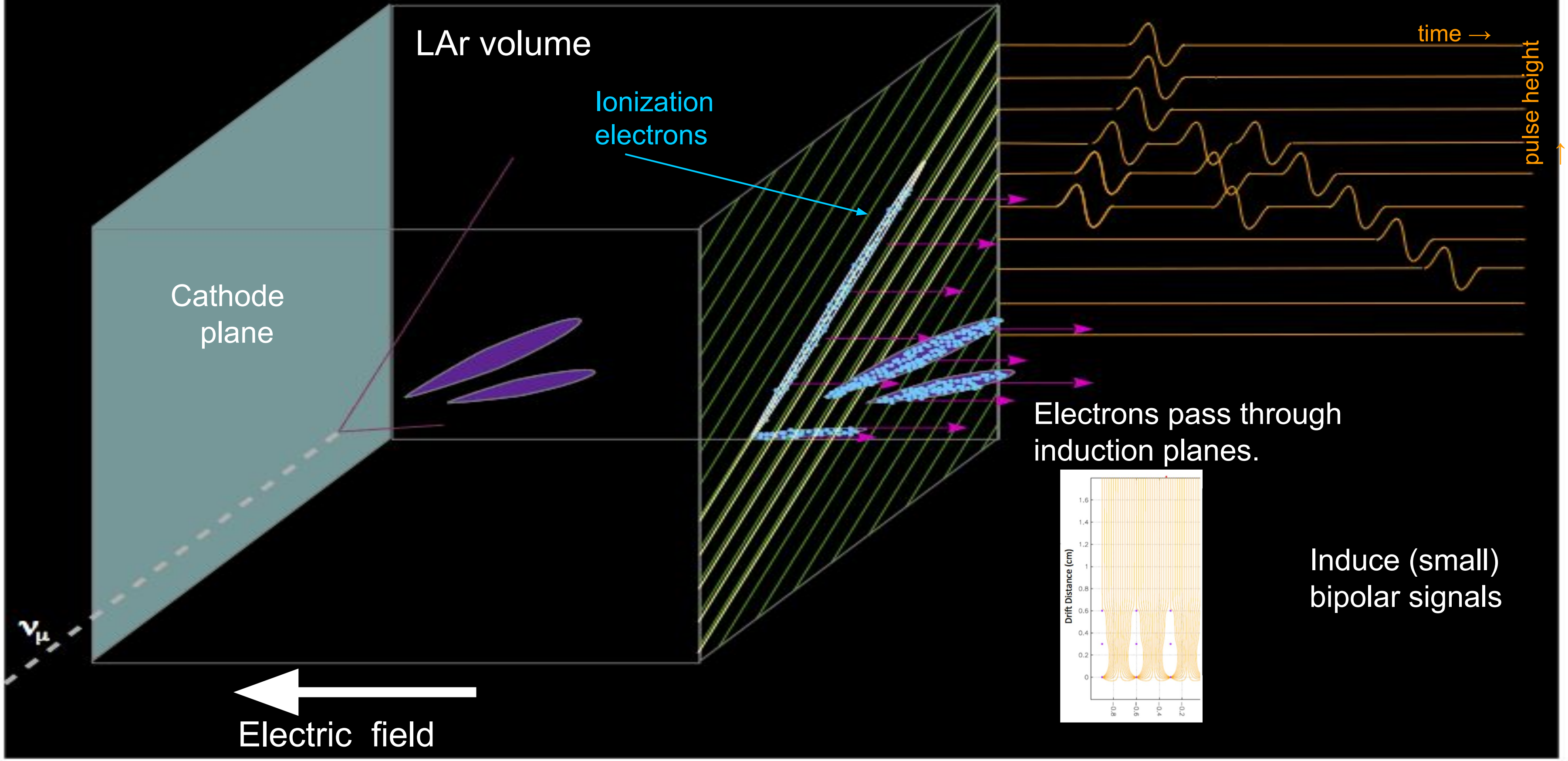
Operation of single-phase LAr TPC



Electrons pass through induction planes.

Induce (small) bipolar signals

Operation of single-phase LAr TPC



LAr volume

Ionization electrons

Cathode plane

time →

pulse height ↑

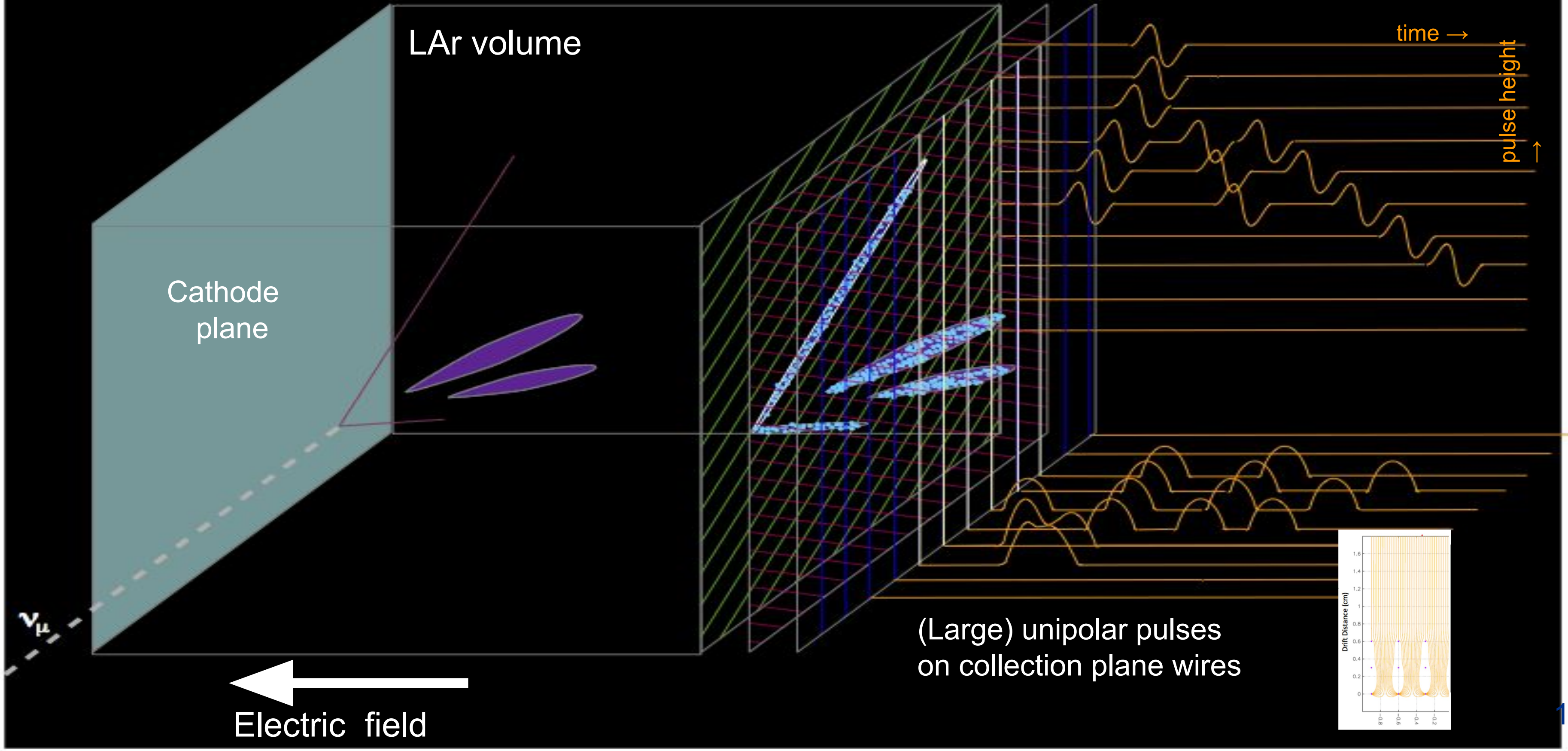
Electrons pass through induction planes.

Induce (small) bipolar signals

ν_μ

Electric field

Operation of single-phase LAr TPC

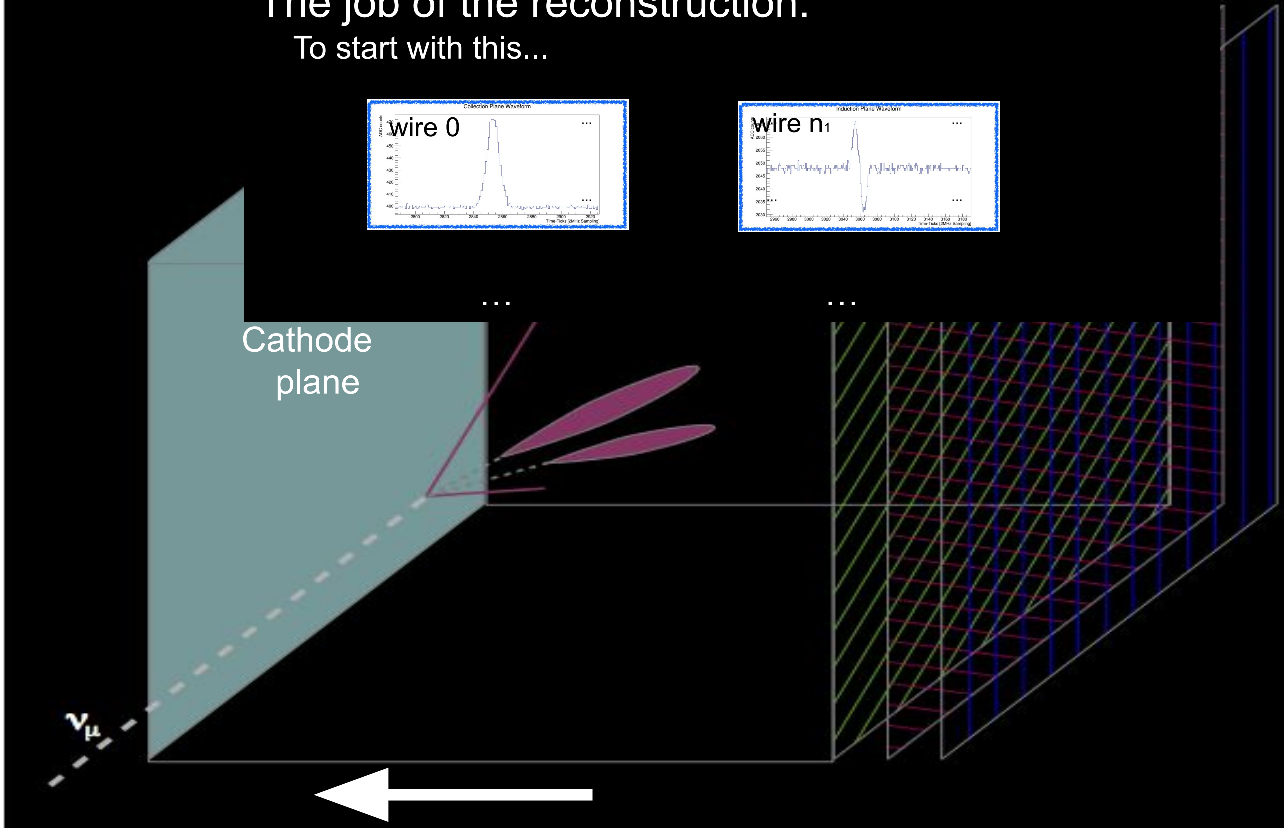


(Large) unipolar pulses
on collection plane wires

Operation of single-phase LAr TPC

The job of the reconstruction:

To start with this...

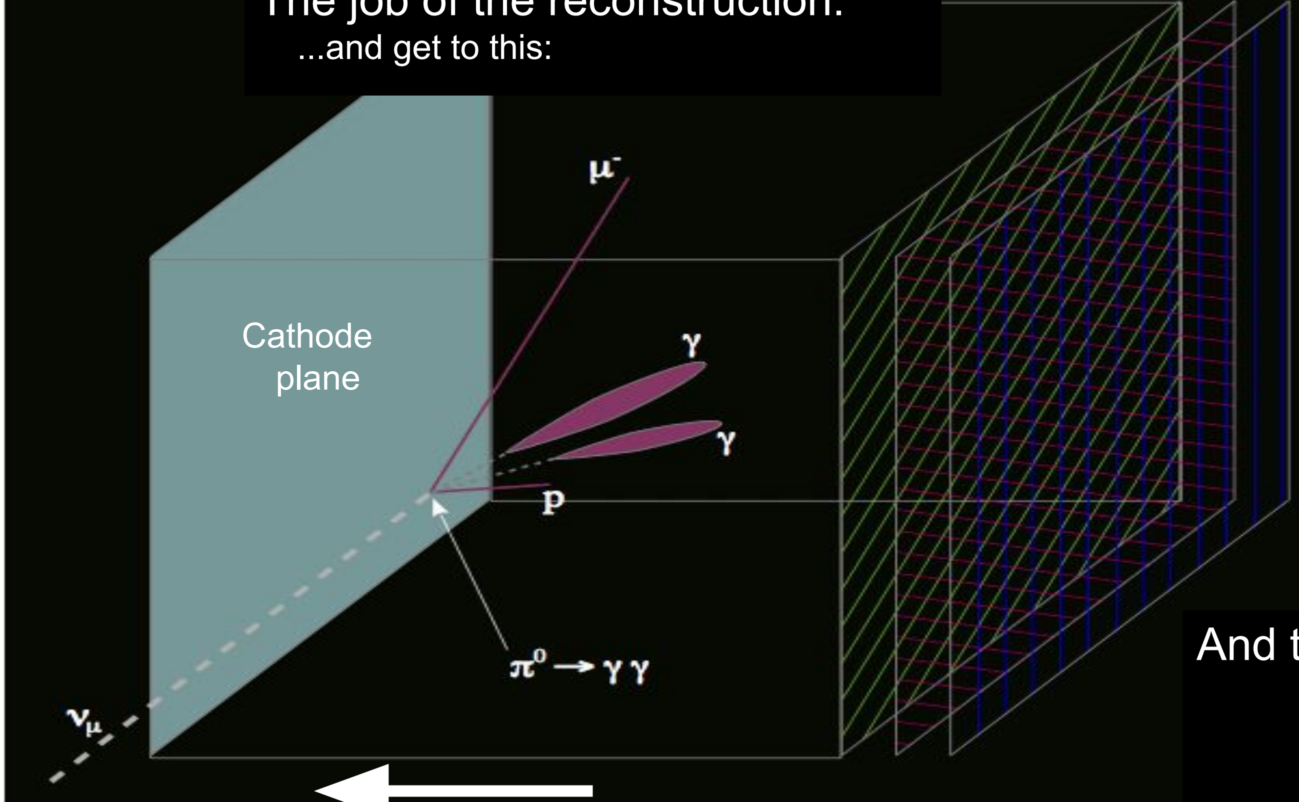


Electric field

Operation of single-phase LAr TPC

The job of the reconstruction:

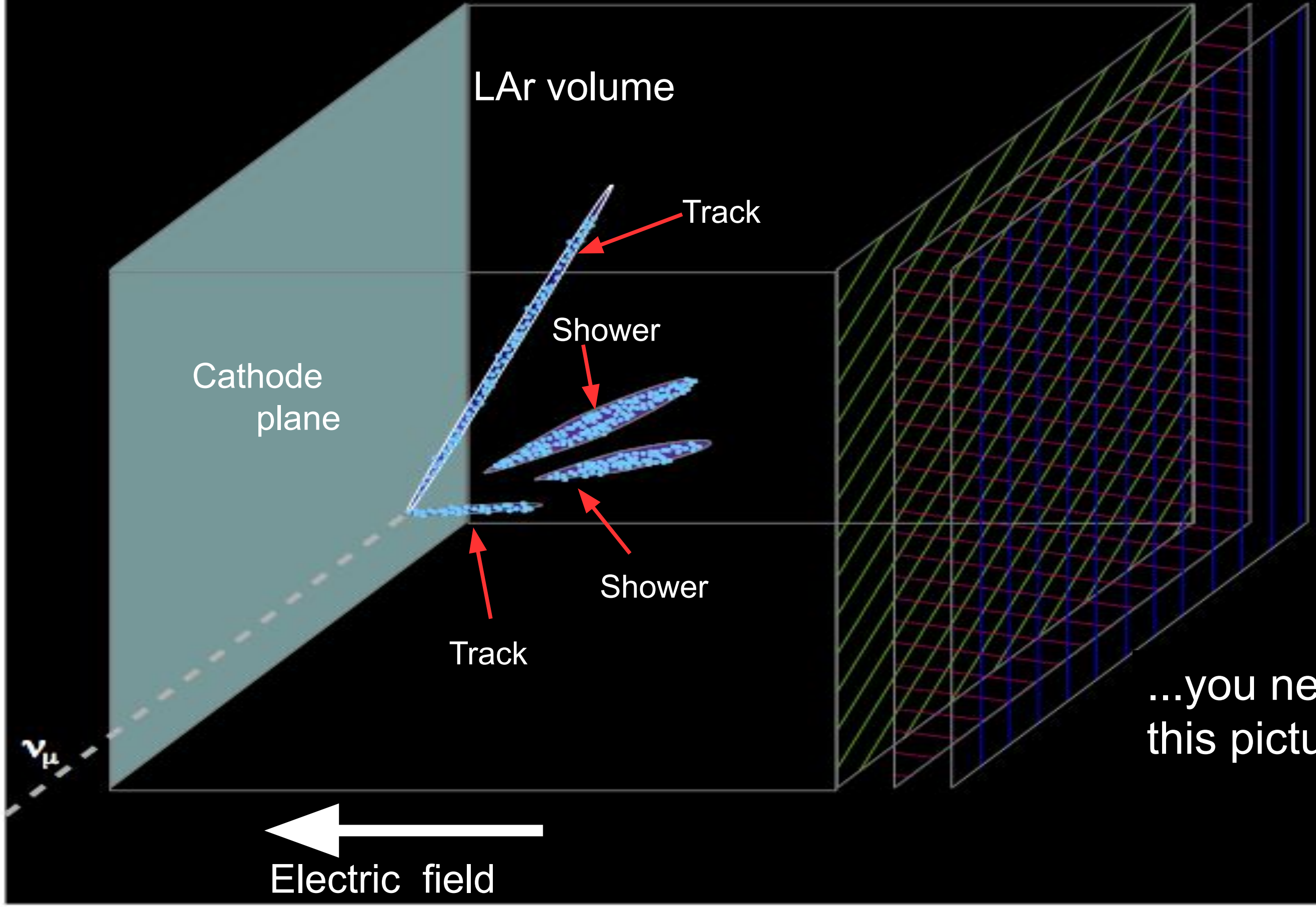
...and get to this:



Electric field

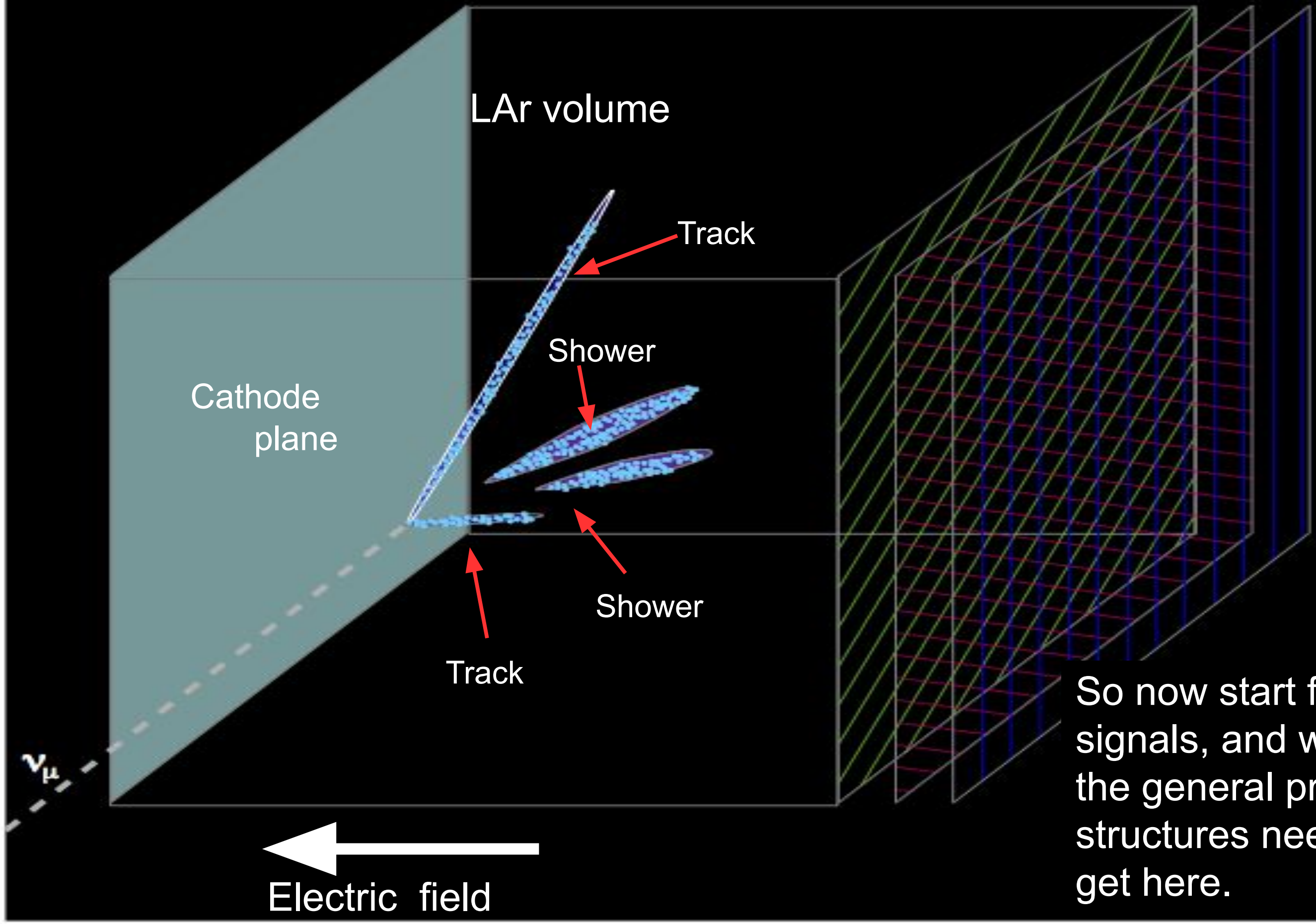
And to get here...

Operation of single-phase LAr TPC



...you need to reconstruct this picture.

Operation of single-phase LAr TPC



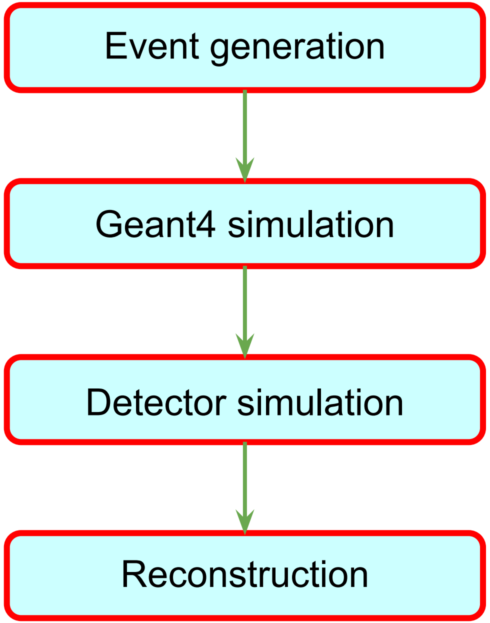
So now start from the raw signals, and walk through the general process, data structures needed to get here.

Simulation and reconstruction in LArSoft

What does LArSoft do? And what is in it?

Provides tools to carry out simulation, reconstruction and analysis of LArTPC data. (Note, analysis uses the output of any of the steps in the workflow, but a discussion of analysis is beyond the scope of this material.)

- Consider for instance, **an event generation, detector simulation, reconstruction workflow**



A general generation – simulation – reconstruction workflow

General generation-simulation-reconstruction workflow

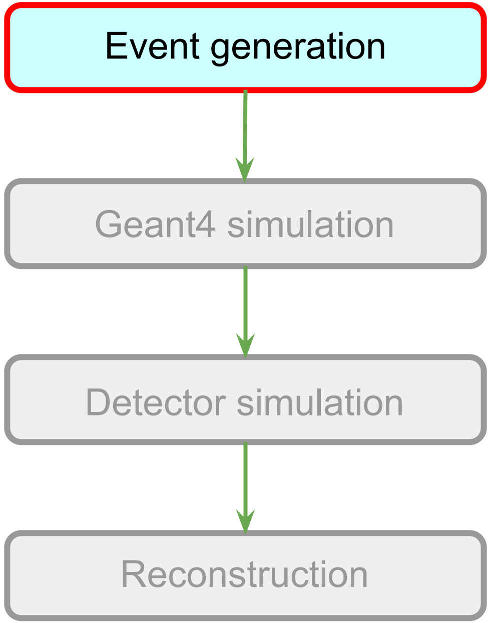
Event generators

- Genie: GENIEGen module
  - Interfaces to Genie neutrino event generator
  - larsim/larsim/EventGenerator/GENIE/
  - See genie.fcl in that directory
  - More documentation on the NuTools wiki page,
  - https://cdcvs.fnal.gov/redmine/projects/nutools/wiki

- Single particles: SingleGen module
  - larsim/larsim/EventGenerator

- Cosmic ray generators: CORSIKA, CRY
  - larsim/larsim/EventGenerator

Others available via indirect common data exchange format, e.g., NuWro



Event generation

Geant4 simulation

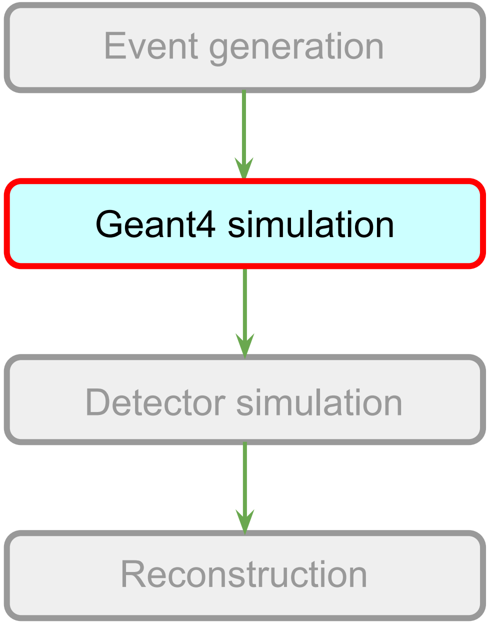
Detector simulation

Reconstruction

General generation-simulation-reconstruction workflow

Geant4 detector simulation

- Particle propagation simulation
- Models energy depositions in the detector
  - Rich, configurable models of particle interactions, optical properties (including detailed index of refraction, reflectivity, etc.)
  - Can perform optical simulation at single photon level
- The only simulation currently integrated with LArSoft



Event generation

Geant4 simulation

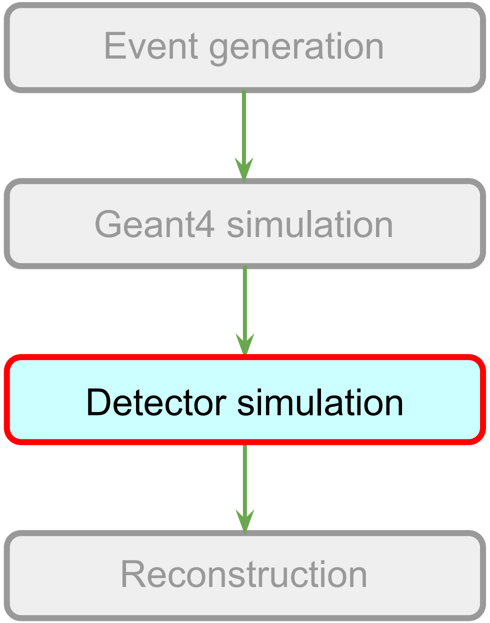
Detector simulation

Reconstruction

General generation-simulation-reconstruction workflow

A separate workflow in itself

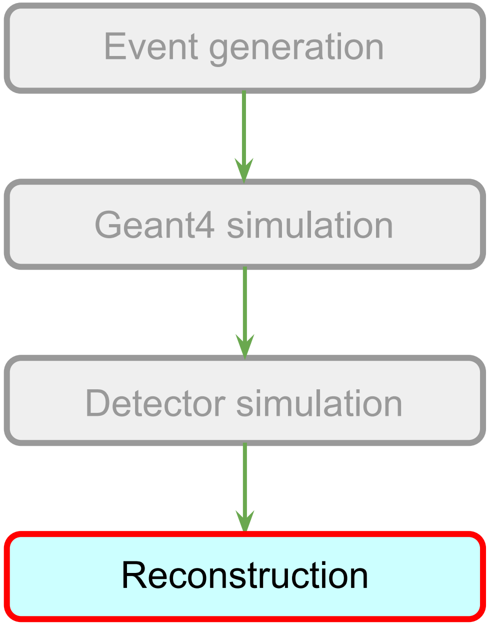
- Factorized into the following steps (implemented as separate modules / partly combined in WireCell)
  - Ionization and scintillation light modeling from energy depositions
  - Drift electron simulation
  - Anode region simulation, signal induction and noise modeling, digitization
  - Photon transport and detection model, including “S2 light” simulation for dual-phase detectors
  - Optical signal induction, noise modeling and digitization



General generation-simulation-reconstruction workflow

Three major paradigms, each with its own variants, modules, workflows

- 2D clustering and view matching
  - Pandora multi-algorithm approach
  - TrajCluster 2D
- Image processing / deep learning techniques
  - Pixel-level track/shower tagging from 2D images (code not yet fully available)
  - Hit-based track/shower discrimination
- 3D imaging
  - Wire-cell: tomographic charge matching across wire planes in time slices
  - TrajCluster3D / Cluster3D: time / charge matching across wire planes using hits.



Event generation

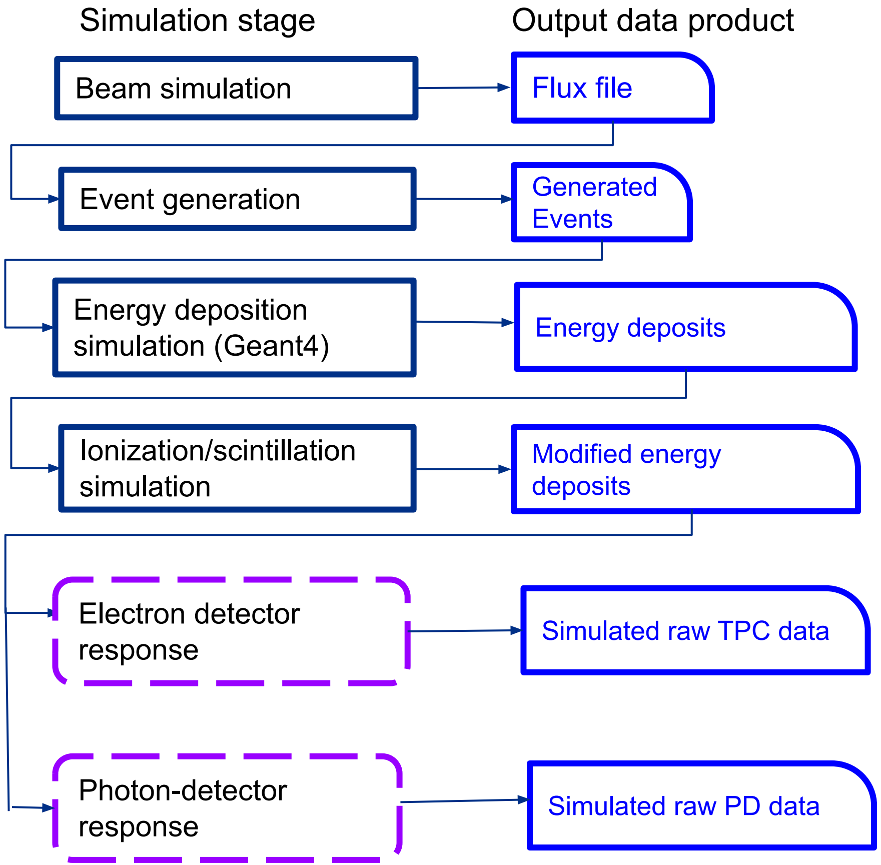
Geant4 simulation

Detector simulation

Reconstruction

Detailed simulation workflow in LArSoft

Simulation workflow

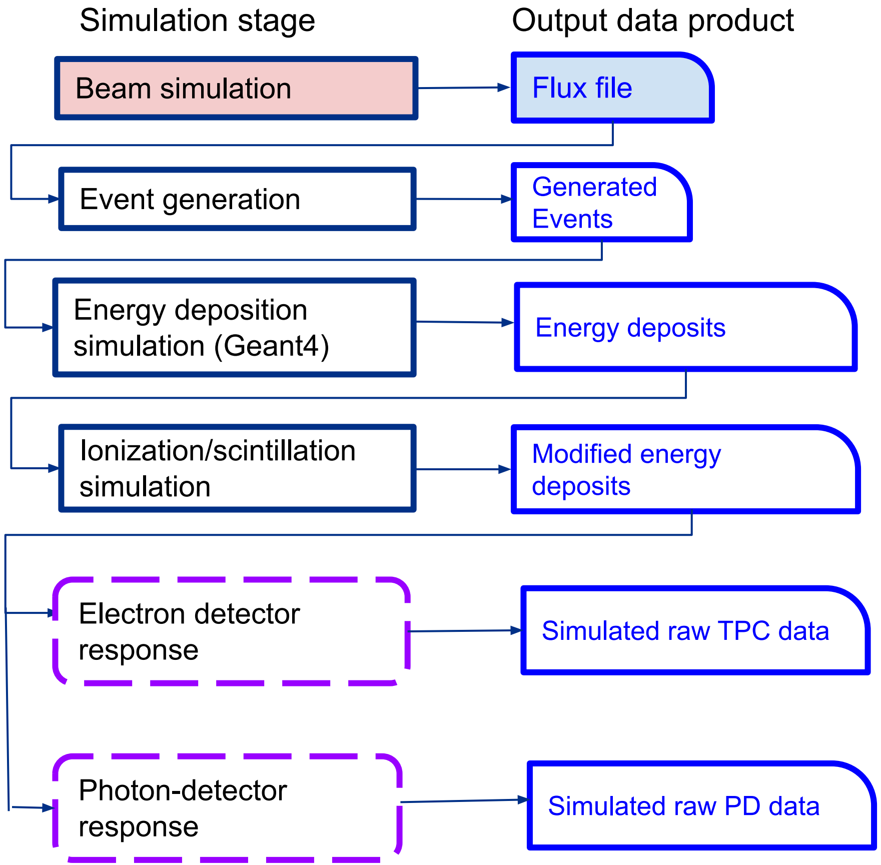


Typically run as at least three separately phases:

- “Beam” simulation
- Event generation
- Detector simulation and response

The detector simulation and response can also be run in several phases, as we will show

Simulation workflow

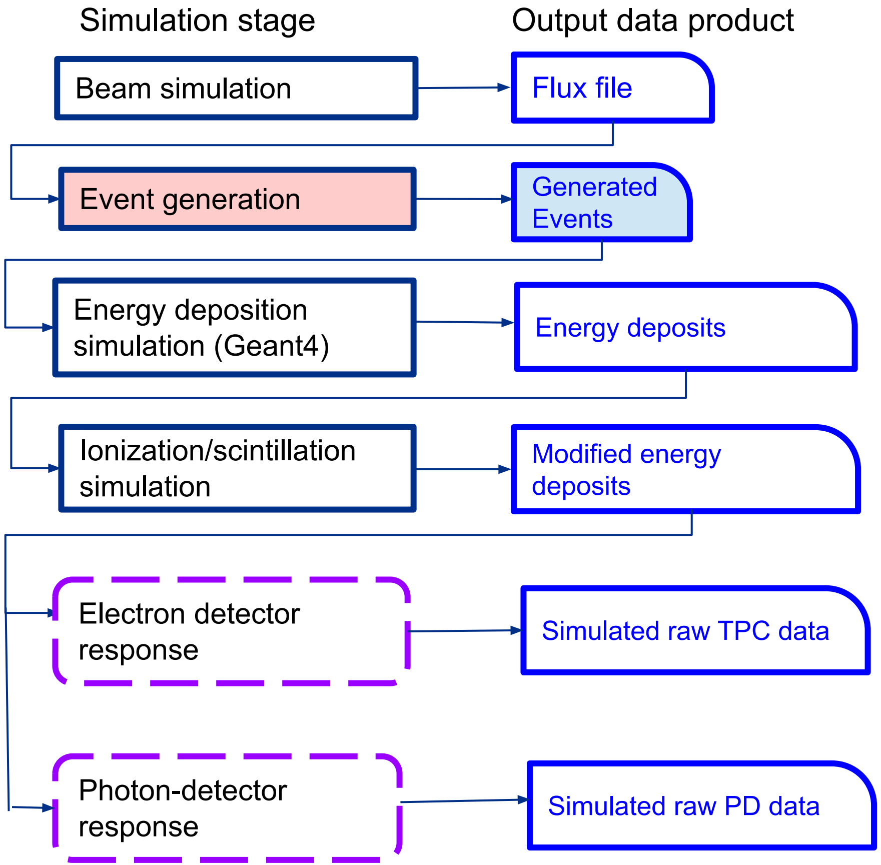


Beam simulation:

- Generates neutrino flux hitting the detector
- Simulated sources can include accelerator, sun, astrophysical sources, KDAR sources, etc. (so not strictly from accelerator beams)

The beam simulation is external to LArSoft

Simulation workflow



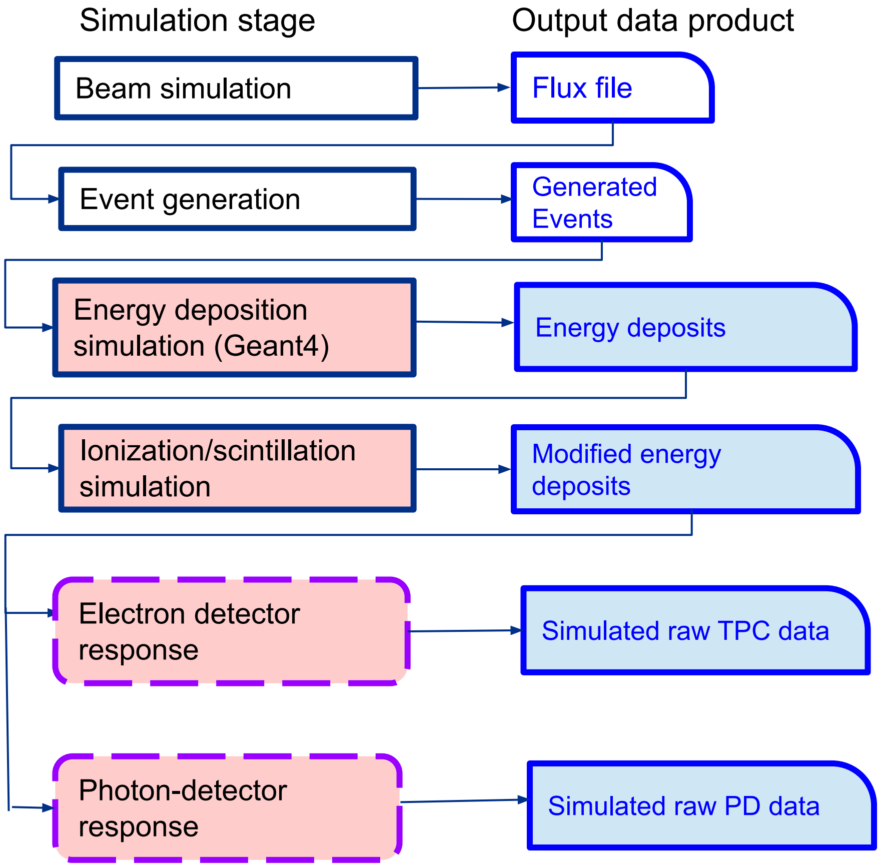
Event generation

- Produces final state secondaries from neutrino interactions within the detector based on input flux description
- For proton decays and radiologicals, just generates decay signatures
- Output is list of final state particles in simb::MCTruth

Lots of options available for the event generator!!

Can run different generators using the same flux files as input.

Simulation workflow

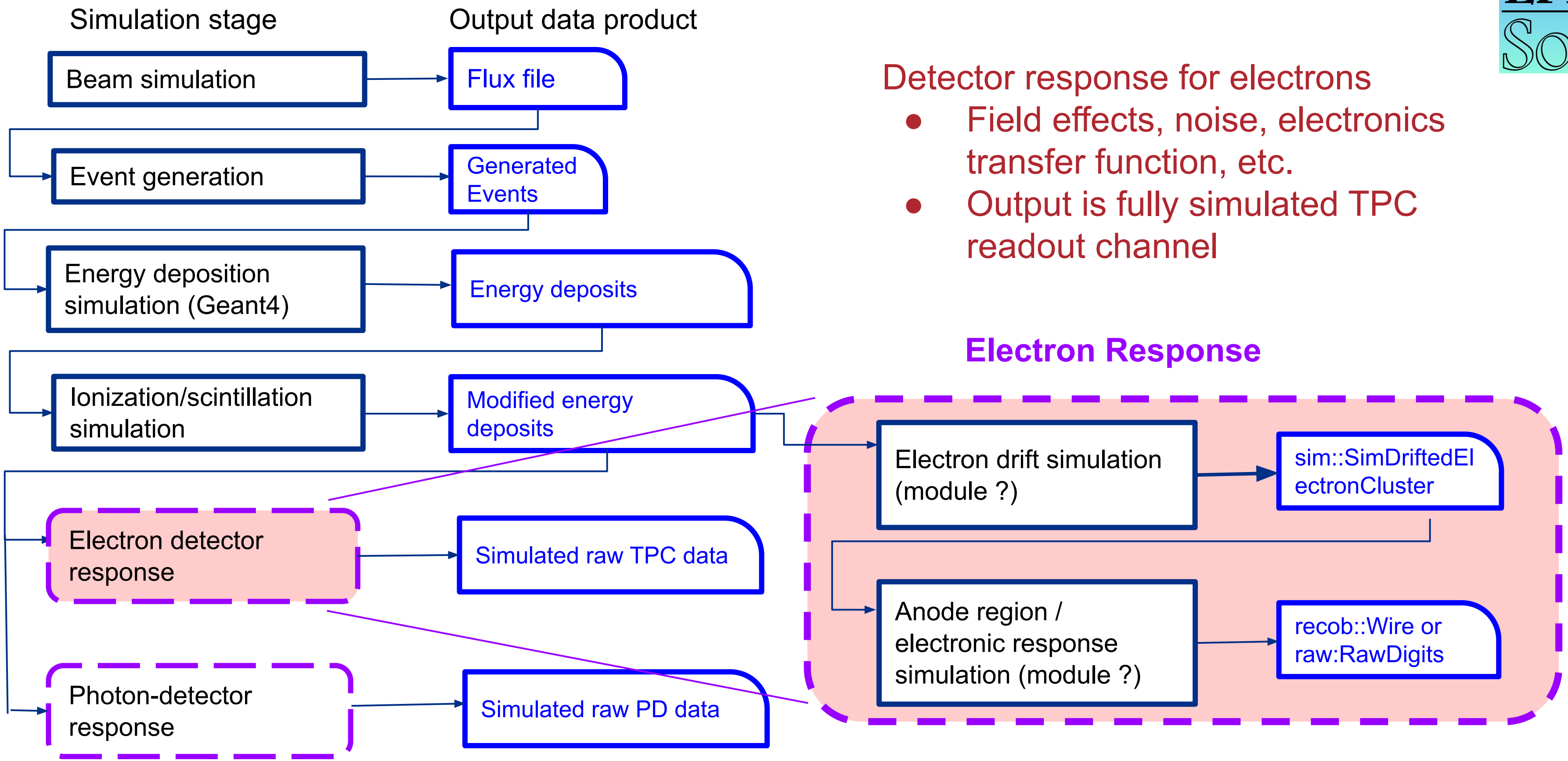


Detector simulation and response

- Given MCTruth, performs all steps necessary to produce simulated output waveforms from detector
- Output waveforms typically post-noise reduction, and field / electronics response deconvolution

Detector response is further factored into separate steps

Simulation workflow

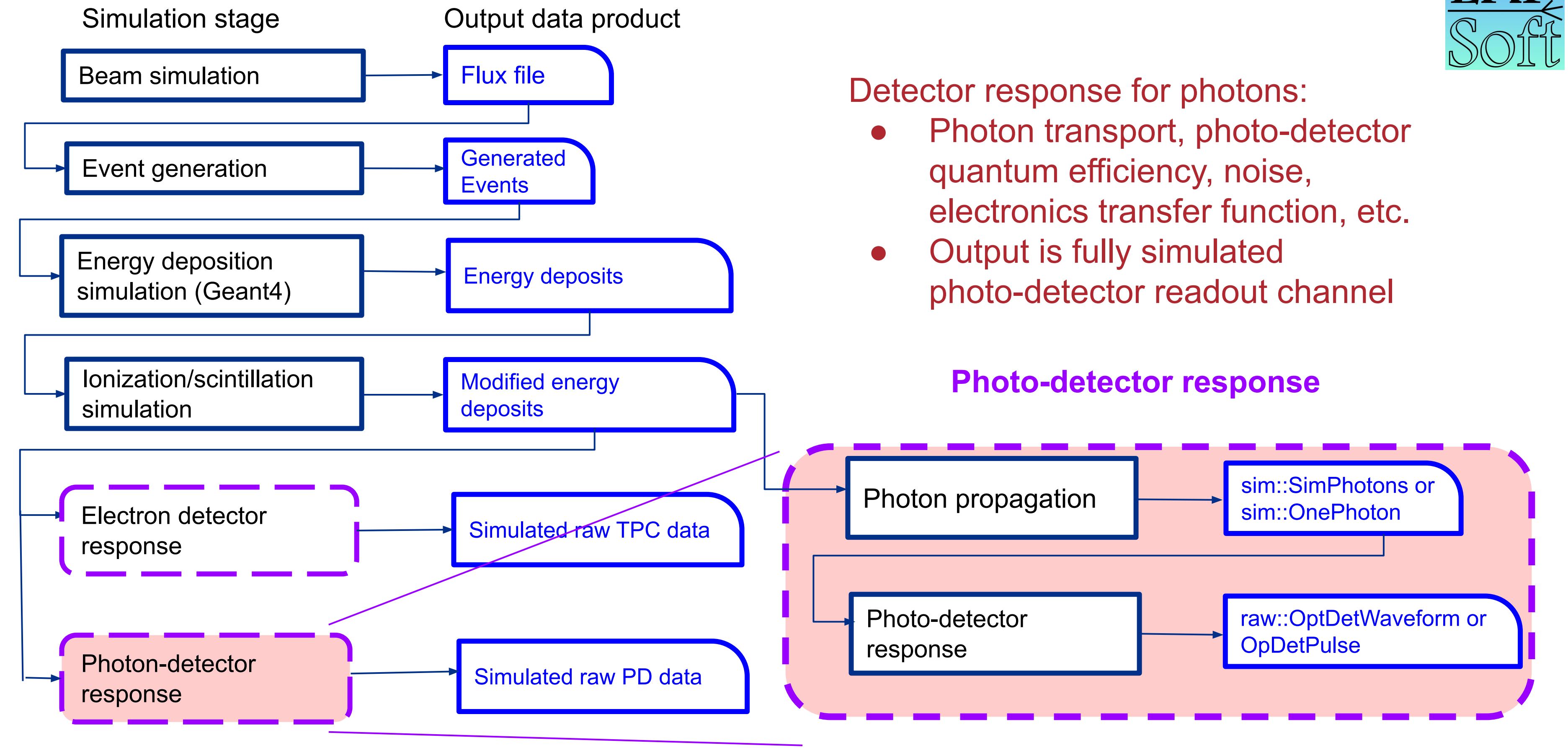


Simulation stage
Output data product
Beam simulation
Flux file
Event generation
Generated Events
Energy deposition simulation (Geant4)
Energy deposits
Ionization/scintillation simulation
Modified energy deposits
Electron detector response
Simulated raw TPC data
Photon-detector response
Simulated raw PD data
Electron Response
Electron drift simulation (module ?)
sim::SimDriftedElectronCluster
Anode region / electronic response simulation (module ?)
recob::Wire or raw:RawDigits

Detector response for electrons

- Field effects, noise, electronics transfer function, etc.
- Output is fully simulated TPC readout channel

Simulation workflow



Detector response for photons:

- Photon transport, photo-detector quantum efficiency, noise, electronics transfer function, etc.
- Output is fully simulated photo-detector readout channel

Photo-detector response

Design principles and coding practices

LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algorithm code
3. Use of standardized algorithm interfaces
4. Modularity
5. Design / write testable units of code
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algorith
3. Use of standardized algorithm interfa
4. Modularity
5. Design / write testable units of code
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

The foundation of the code sharing regime

Possible because the nature of LArTPCs allows for the use of many common interfaces, with differences expressed as differences in configuration

Will expand on detector interoperability later...

LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algorithm
3. Use of standardized algorithm interfaces
4. Modularity
5. Design / write testable units of code
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

- Critically important
- Allows use of LArSoft algorithm code outside of art, such as:
  - Lightweight analysis frameworks
    - Gallery, LArLite, ...
  - Specialized development / debugging environments
- Allows a future migration to another production framework, should that be needed

LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algorithm
3. Use of standardized algorithm interface
4. Modularity
5. Design / write testable units of code
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

- Encapsulate algorithms, configuration, tools and utilities into a layer that is independent of the *art* framework (eg, no art::Handle<> in algorithms)
- Requires adherence to proper coding practices and physics designs
  - Use modules to interact with art::Event, obtain services, etc.
  - Construct services such that the service (the class registered with art) handles art callbacks, but delegates all the work to a "provider" that knows nothing about art
  - Pass event data and service providers to algorithm code

LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algorithm code
3. Use of standardized algorithm interfaces
4. Modularity
5. Design / write testable units of c
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

Provides a means to hide detector-specific details behind common interfaces

Also allows layering of algorithms to build sophistication

LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algorithm code
3. Use of standardized algorithm interfaces
4. Modularity
5. Design / write testable units of code
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

Just good coding practice…

Build sophistication by applying algorithms in a layered, iterative structure.

LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algorithm
3. Use of standardized algorithm interfac
4. Modularity
5. Design / write testable units of code
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

Ensures that code operates as intended
Simplifies code integration

LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and al
3. Use of standardized algorithm i
4. Modularity
5. Design / write testable units of c
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

So that other people understand what your code is supposed to do, and how to use it

So that you know what your code is supposed to do and how to use six months after you wrote it…

Use Doxygen markup in source code comments

Include at a minimum the purpose of the file, how it is used, pre-requisites, assumptions, etc.

LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algori
3. Use of standardized algorithm inter
4. Modularity
5. Design / write testable units of code
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

New! (relatively)

Expect multi-threading to play an increasingly important role

- To help control scaling of memory usage
- To adapt to the evolving computing landscape

LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algorithm code
3. Use of standardized algorithm interfa
4. Modularity
5. Design / write testable units of code
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

Ensures stability of the development environment

Allows rapid development cycles

Simplifies release management

LArSoft design principles and coding practices

The basic philosophies and rules that underlie code sharing in core LArSoft code

1. Detector interoperability
2. Separation of framework and algorithm code
3. Use of standardized algorithm interfaces
4. Modularity
5. Design / write testable units of code
6. Document code in the source
7. Write code that is thread safe
8. Continuous integration

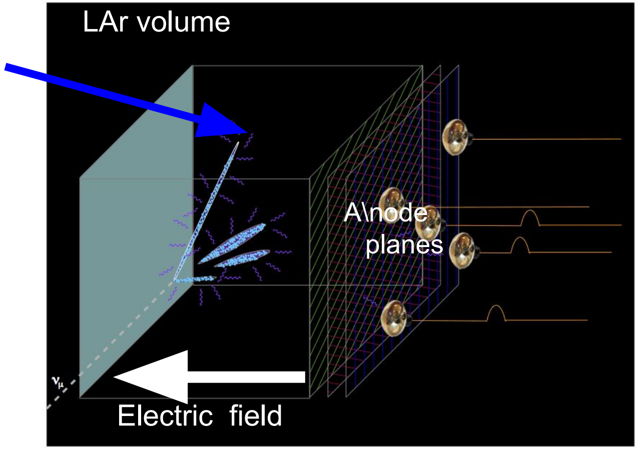
Additions and changes will be made as needed to adapt to changes in the computing landscape, or to better support code sharing

LArSoft Detector Interoperability (#1 Design Principle)

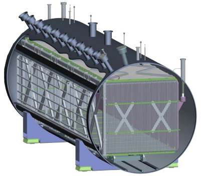
Detector interoperability of LArSoft software

- The cornerstone of LArSoft design and architecture
- Rests on common features of LAr TPC geometry, physics, data

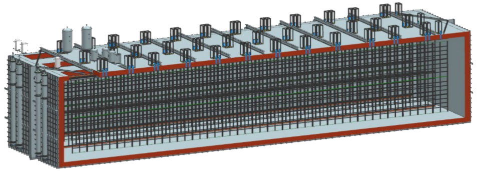
Active volume of LAr with uniform E-field...



LArIAT
0.4m x 0.47m x 0.9m



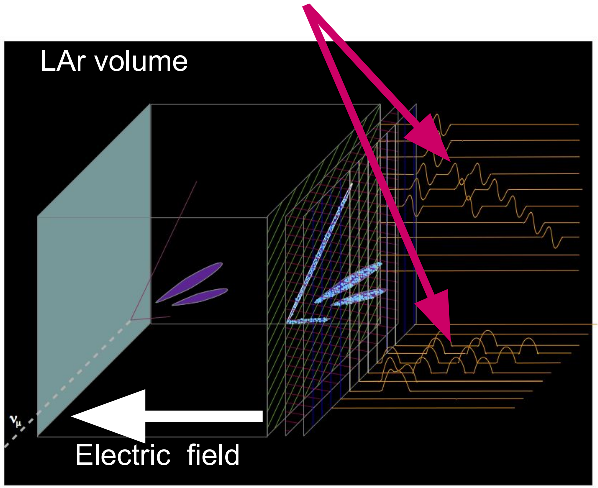
MicroBooNE
2.2m x 2.5m x10m



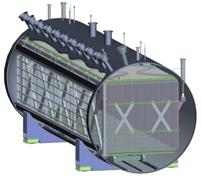
DUNE far detector module
18m x 19m x 66m

Detector interoperability of LArSoft software

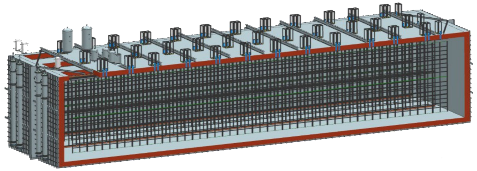
...Digitized waveforms in multiple views
induced by motion or collection of ionization...



LArIAT
0.4m x 0.47m x 0.9m



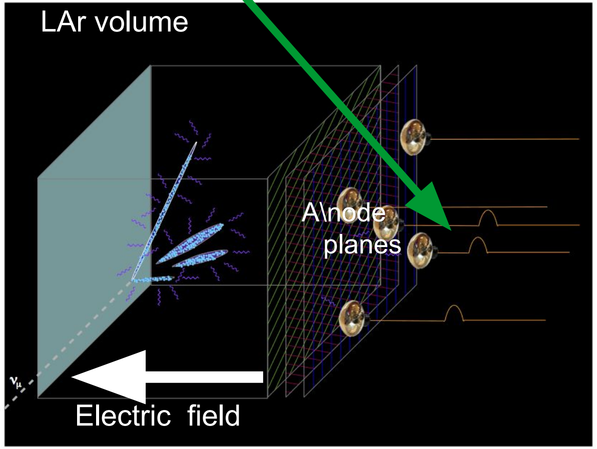
MicroBooNE
2.2m x 2.5m x10m



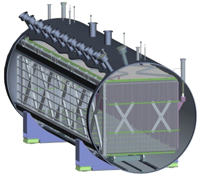
DUNE far detector module
18m x 19m x 66m

Detector interoperability of LArSoft software

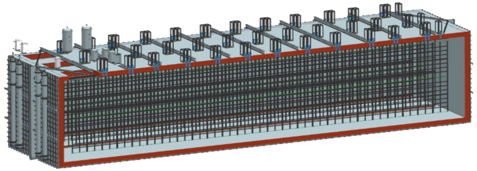
...Digitized waveforms from of detected scintillation light from multiple photo-detectors...



LArIAT
0.4m x 0.47m x 0.9m



MicroBooNE
2.2m x 2.5m x10m

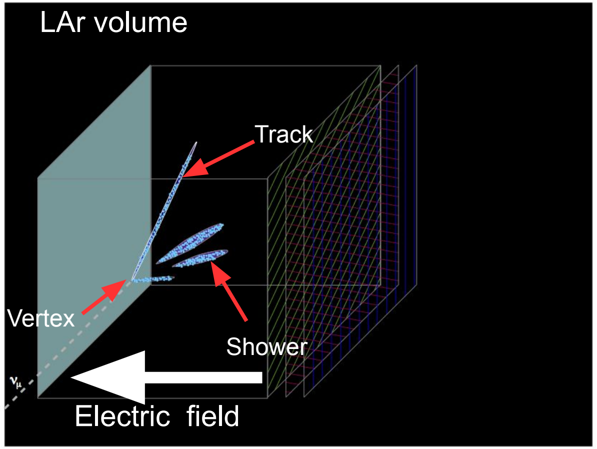


DUNE far detector module
18m x 19m x 66m

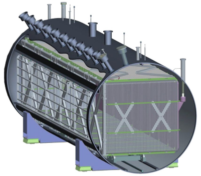
Detector interoperability of LArSoft software

...reconstructed signals, 2D and 3D objects, measurements of physical properties such as range and dE/dx

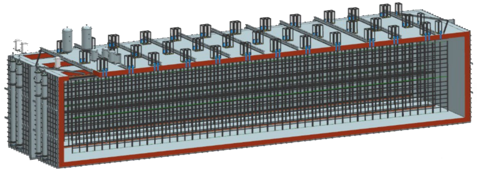
Allows definition of shared data structures, interfaces, workflow stages, and ultimately, shared algorithms, physics tools, utilities



LArIAT
0.4m x 0.47m x 0.9m



MicroBooNE
2.2m x 2.5m x10m



DUNE far detector module
18m x 19m x 66m

Detector interoperability of LArSoft software

- Detector and time-dependent conditions data
  - Geometry: use a generic interface to obtain geometry information
    - Facilitated by
      - Detector and data IDs defined at all levels
      - Creation of tools for generic loops over geometric elements
      - Strict avoidance of implicit geometrical assumptions in the code
  - Similarly, use shared interfaces to calibration, electric field maps, conditions information, etc.
    - Implementations differ by back-end database schemas, other detector-specific details

Detector-specific elements

- Specify handling of many detector-dependent details via configuration (FHiCL files)
  - Input geometry description
  - Source for generic detector properties, LAr conditions and properties
  - Back-end for calibration data
  - Source and back-end for photon transport / detection maps
  - Source and back-end for electric field map
  - Etc.

- Detector-specific implementations currently required for
  - Raw data noise removal and signal processing
  - Electronics response in simulation and reconstruction
  - Simulation of raw data digitization

General disclaimer:
In examining the code, you may note that only a portion currently adheres to these principles.

We strongly encourage people to adopt these practices for all new code.

Separation of framework and algorithm (#2 Design Principle)

Separation of framework and algorithm

Achieve separation by:

- Adhering to certain coding practices
  - art service design pattern
  - Restrictions on art module code

  in order to create an art-independent layer for algorithms, configuration, (*art*) tools and utilities

- Factoring I/O, *art* event data model (canvas) out of the art framework
  - Event data model (via canvas) is available for use in the “art-independent” code
  - Note that FHiCL and message service do not depend on art, so can also be included directly in “art-independent” code

Separation of framework and algorithm

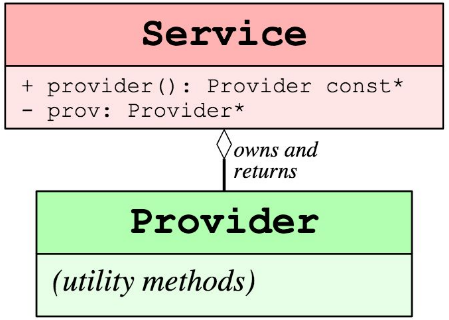
art **service design pattern**

A LArSoft service is a class, with a single instance managed by the framework, that performs an operation. A service is used by LArSoft algorithms and *art* modules.

To be used in algorithm code, LArSoft services are factorized into two parts:

1. A “service provider” with no dependence on *art* that does the work of the service
   a. Algorithm code interacts with the provider
   b. The provider is passed in to algorithms
2. An “*art* service” that interfaces the provider with the *art* framework
   a. This is the part that is registered with *art* at run-time

This factorization model allows service providers used and tested without pulling in the *art* framework, and to be used in art-unaware environments



Separation of framework and algorithm

Examples of LArSoft Services with this structure

- Geometry
- LAr properties
- Detector properties
- Access to databases for calibrations, channel status, etc.
- Photon visibility (part of predicting photo-detector response)
- ...

To write services from scratch, one can start with the examples in larexample repository

Separation of framework and algorithm

Restrictions on art module code

Treat modules as interfacing algorithms to the framework

An algorithm is a piece of code that:

- performs one single task, or a set of algorithms
- In principle, can be a component of many execution paths, and used in multiple modules

(cont on next slide)

Separation of framework and algorithm

A LArSoft algorithm must be able to perform its task using only:

- LArSoft data products and their associations (input and output data)
- Service providers
- FHiCL parameter sets
- Calls to message_service allowed

Write art modules that:

- Get configuration data from ParameterSet passed to module
- Get data products from, and put them into the event
- Get service instances
- Create algorithm instances (if they are classes)
- Call algorithm methods, passing data products, service providers, ParameterSet(s)

Separation of framework and algorithm

art

“*art* is the event-processing framework developed and supported by the Fermilab Scientific Computing Division (SCD).

“The *art* framework is an application used to build scientific programs by loading science algorithms, provided as plug-in modules; each experiment or user group may write and manage its own modules. *art* also provides **infrastructure for** common tasks, such as **reading input, writing output, run-time configuration**, provenance tracking, **message handling** and database access.”

The parts in bold are separate products, and are not formally part of the event-processing framework code

Separation of framework and algorithm

Gallery

gallery provides lightweight access to event data in *art*/ROOT files outside the *art* event processing framework.

gallery is not an alternative framework; rather, it provides a library that can be used to write programs that need to read (but not write) *art*/ROOT files. You must have access to the ROOT dictionaries for the classes in a data file to use that data file. The availability of such dictionaries is provided by the experiments.

gallery is built:

- without the use of EDProducers, EDAnalyzers, etc., thus
- without the facilities of the framework (e.g. callbacks from framework transitions, writing of *art*/ROOT files).

Algorithm code may be called within code that uses Gallery for event access

Separation of framework and algorithm

Canvas

The canvas package is the infrastructure required for providing I/O operations for the full art framework and the lightweight gallery framework. In particular, the ROOT dictionaries art provides for experiments to use are located in canvas.

A tutorial is available at: https://github.com/marcpaterno/gallery-demo

Algorithm code may use Canvas internally to support data product associations

Separation of framework and algorithm

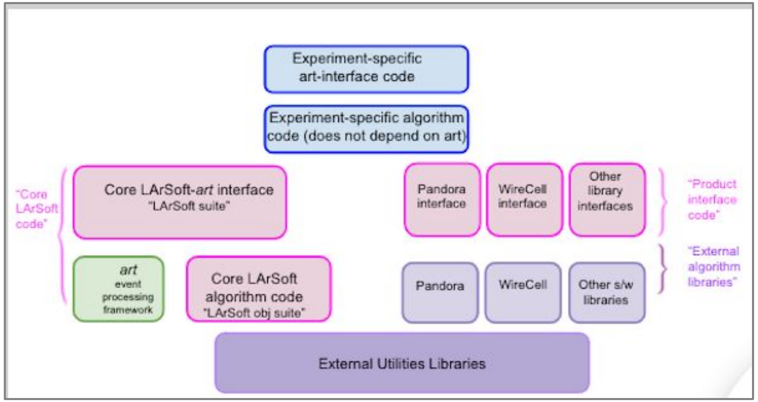
Notes on alternate frameworks

- Properly written LArSoft code can be ported to a new framework by providing a layer of code that can get and put data products (which are just simple classes) in the alternate event record; instantiate and pass service providers, and perform the required functions at state transitions to keep the provider up to date; fill the appropriate ParameterSet(s) from the new source of configuration data; interface as needed with message facility

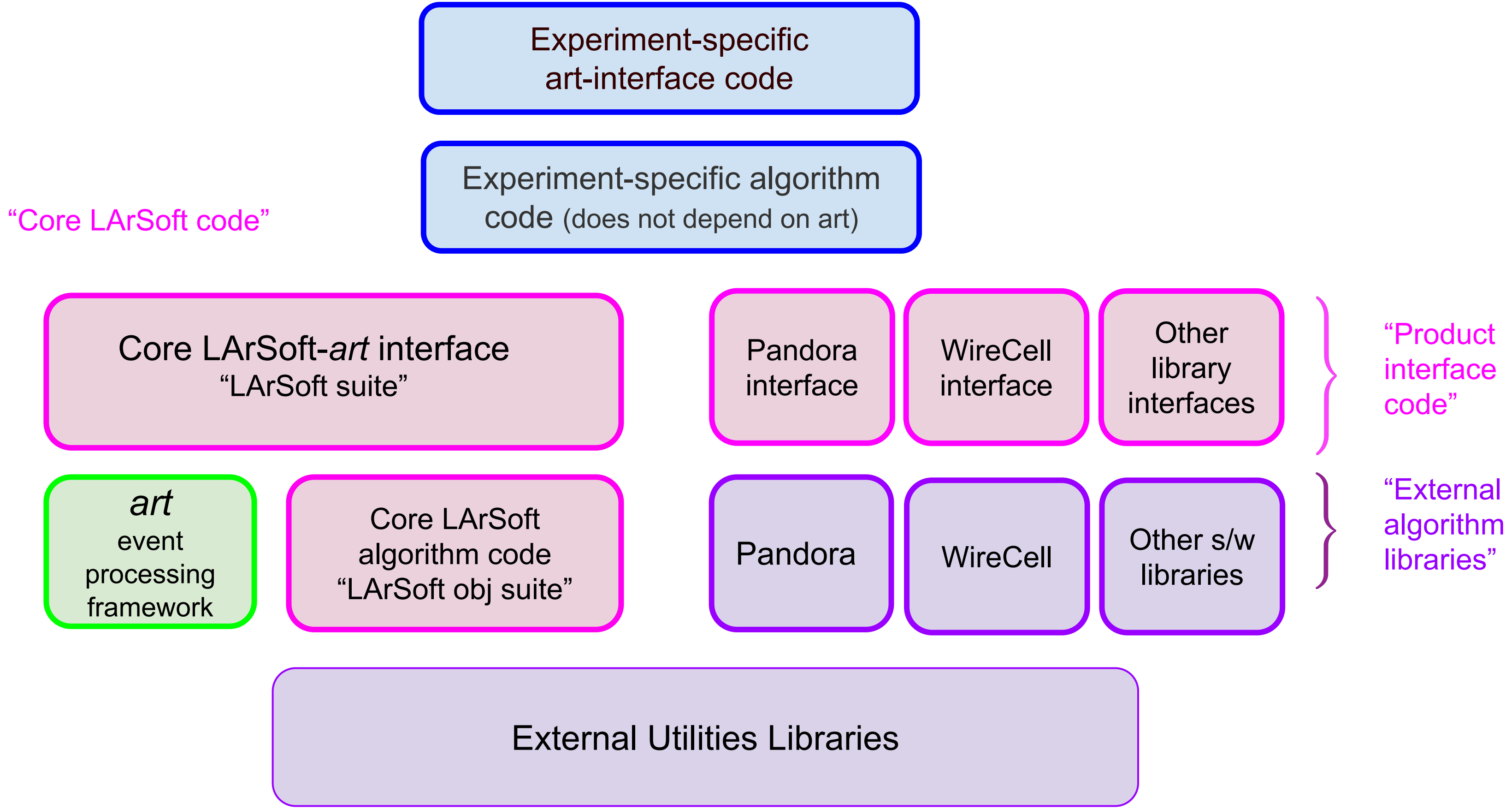
LArSoft physical design

LArSoft code

- Physical design follows from design principles
  - Detector interoperability
  - Separation of algorithm and framework interface code
  - Modularity


Experiment-specific art-interface code
Experiment-specific algorithm code (does not depend on art)
"Core LArSoft code"
Core LArSoft-*art* interface "LArSoft suite"
Pandora interface
WireCell interface
Other library interfaces
"Product interface code"
"External algorithm libraries"
art event processing framework
Core LArSoft algorithm code "LArSoft obj suite"
Pandora
WireCell
Other s/w libraries
External Utilities Libraries

Structural components of LArSoft

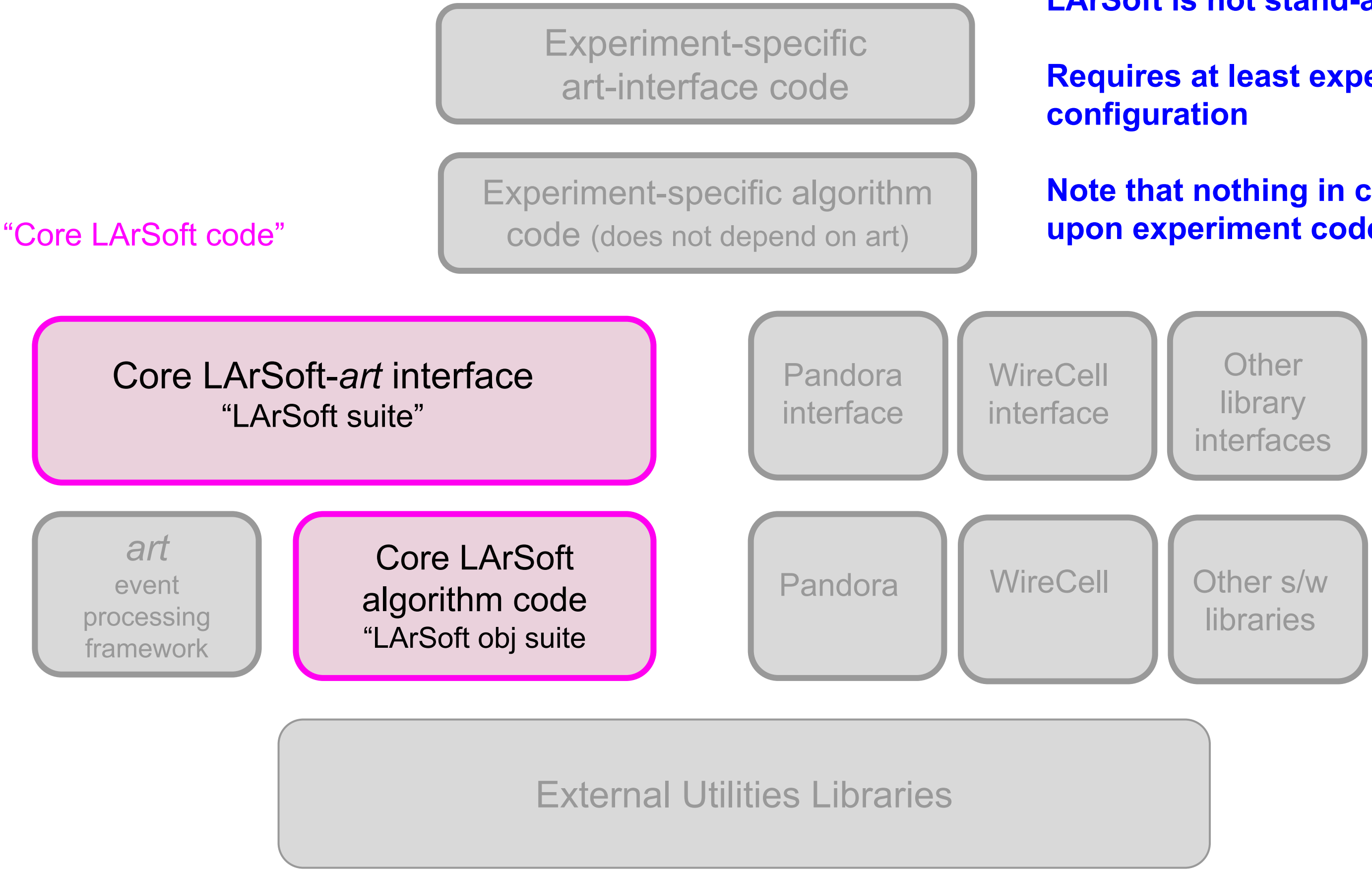


Structural components of LArSoft

LArSoft is not stand-alone code.

Requires at least experiment / detector-specific configuration

Note that nothing in core LArSoft code depends upon experiment code



Experiment-specific art-interface code

Experiment-specific algorithm code (does not depend on art)

"Core LArSoft code"

Core LArSoft-*art* interface "LArSoft suite"

Pandora interface

WireCell interface

Other library interfaces

art event processing framework

Core LArSoft algorithm code "LArSoft obj suite

Pandora

WireCell

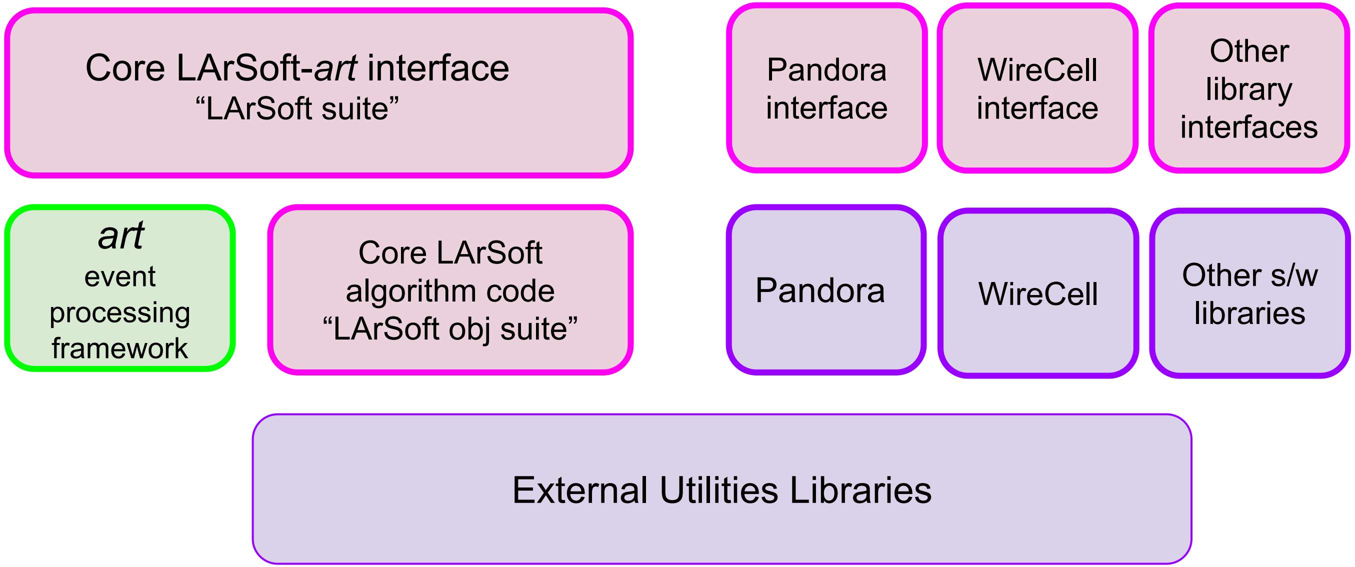
Other s/w libraries

External Utilities Libraries

Conceptual design of LArSoft code

Organizing principle for LArSoft based on a layering of functionality, dependencies

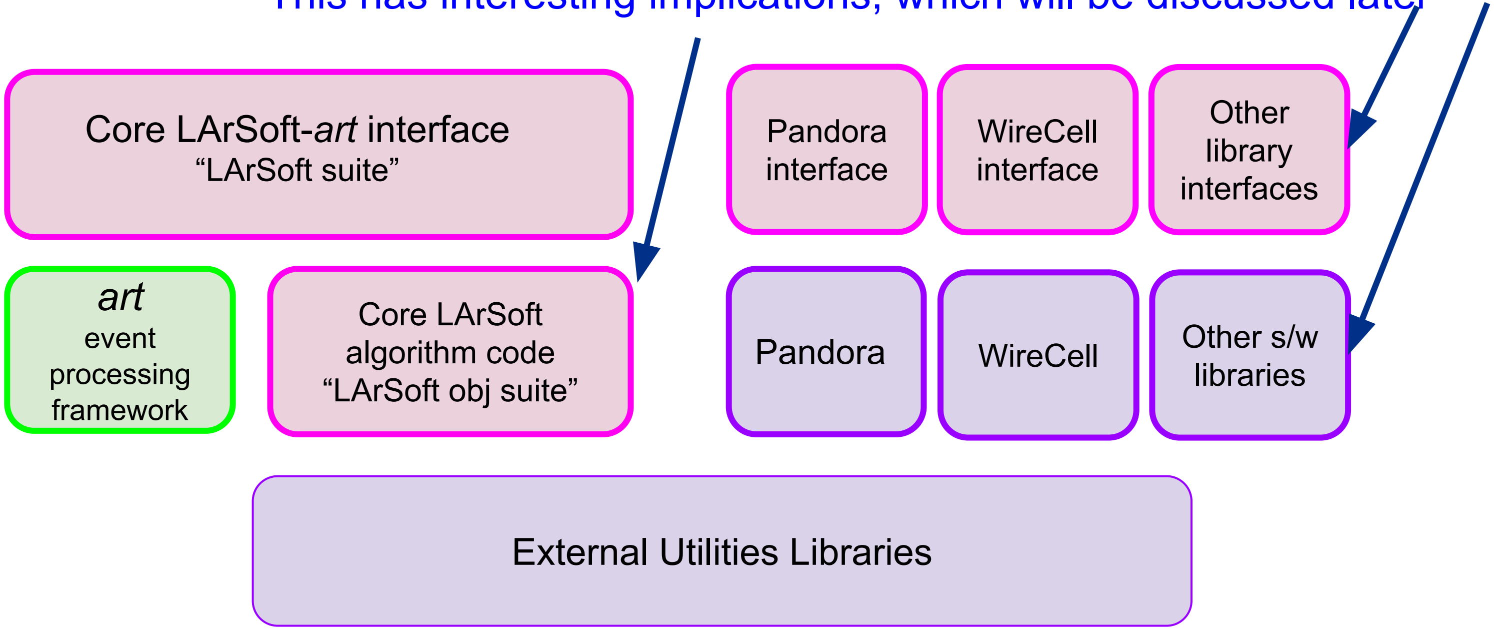
Ideally, layers should only know about the **interface** to the layer **below**



Conceptual design of LArSoft code

Neither LArSoft obj suite nor anything below it knows about or depends on *art*
(though LArSoft obj can use event model (canvas), message facility, FHiCL C++ interface)

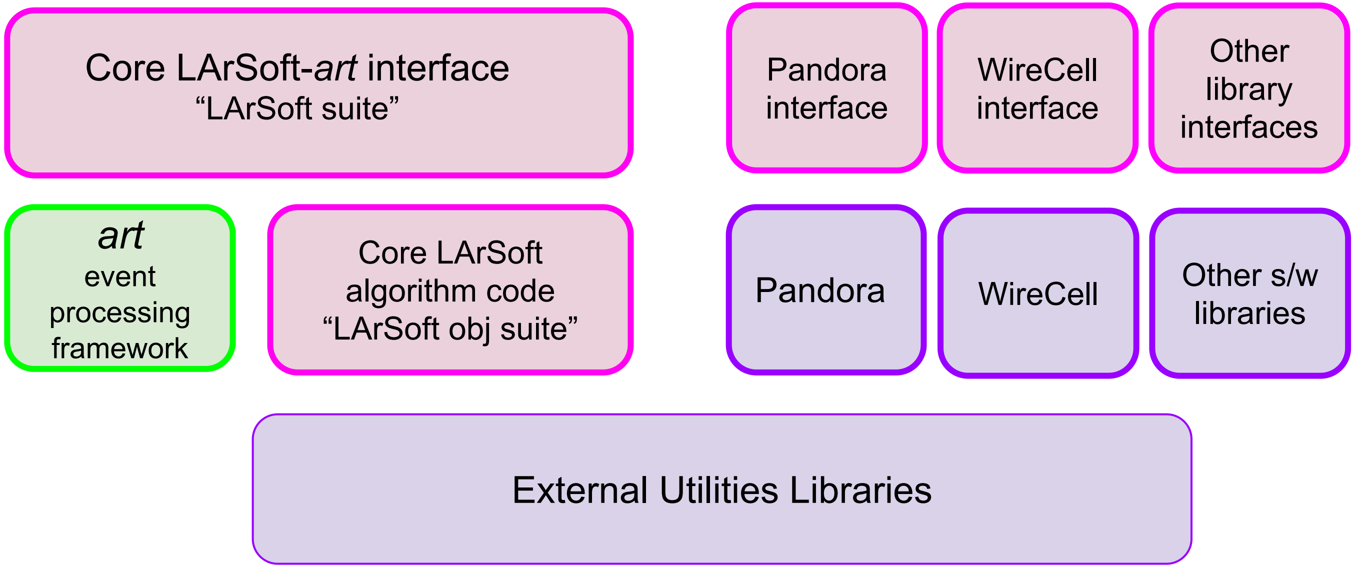
This has interesting implications, which will be discussed later



Conceptual design of LArSoft code

LArSoft built on top of *art* event processing framework.

It is possible to operate with gallery which provides lightweight access to event data in art/ROOT files. Can read (but not write) these files via gallery.



The *art* event processing framework

art
event
processing
framework

Quick art tutorial

- **Reads events from user-specified input source**
- **Executes workflow of tasks as configured via input FHiCL file**
  - Operate on “data products” stored in event records
- **Tasks (algorithms, event filtering, ...) carried out via user-specified “modules” and other “plug-ins”**
  - Dynamically-loaded
  - Can be user-written
  - Configurable via FHiCL files
- **Output data products may be written to output file(s)**

The *art* event processing framework

Three types of plug-ins

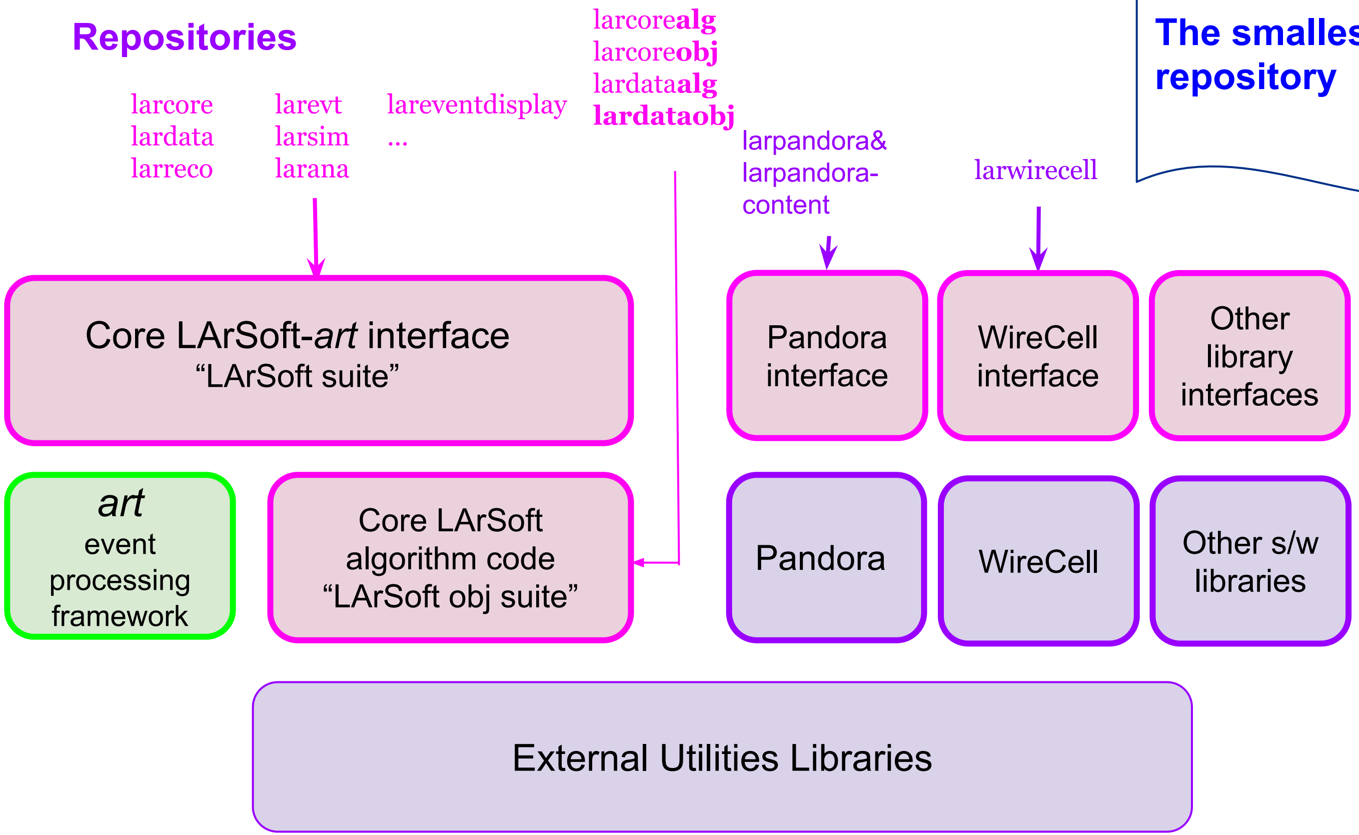
1. **Modules**
   - The basic, scheduled elements within task workflows.
     - *art* calls pre-defined methods at specific times in the event loop
   - Three types
     - Producer: may modify the event
     - Filter: can alter trigger path execution
     - Analyzer: may not modify the event
2. **Services**
   - Classes with global scope that can be accessed within modules.
     - *art* calls registered methods at specific times in the event loop
3. **Tools**
   - Functions or classes with module (or service) scope that have user-specified interface to perform tasks

The *art* event processing framework

More information:

- The art documentation site: resources, detailed tutorials
    - https://art.fnal.gov/
- The art wiki: reference information, coding guidelines, issue tracker
    - https://cdcvs.fnal.gov/redmine/projects/art/wiki
- The FHiCL quick start guide
    - https://cdcvs.fnal.gov/redmine/documents/327
- The FHiCL-cpp wiki: C++ bindings
    - https://cdcvs.fnal.gov/redmine/projects/fhicl-cpp/wiki

Structural components of LArSoft



Repositories

larcore
lardata
larreco

larevt
larsim
larana

lareventdisplay
...

larcore**alg**
larcore**obj**
lardata**alg**
lardataobj

larpandora&
larpandora-content

larwirecell

Core LArSoft-*art* interface
“LArSoft suite”

Pandora interface

WireCell interface

Other library interfaces

art
event processing framework

Core LArSoft algorithm code
“LArSoft obj suite”

Pandora

WireCell

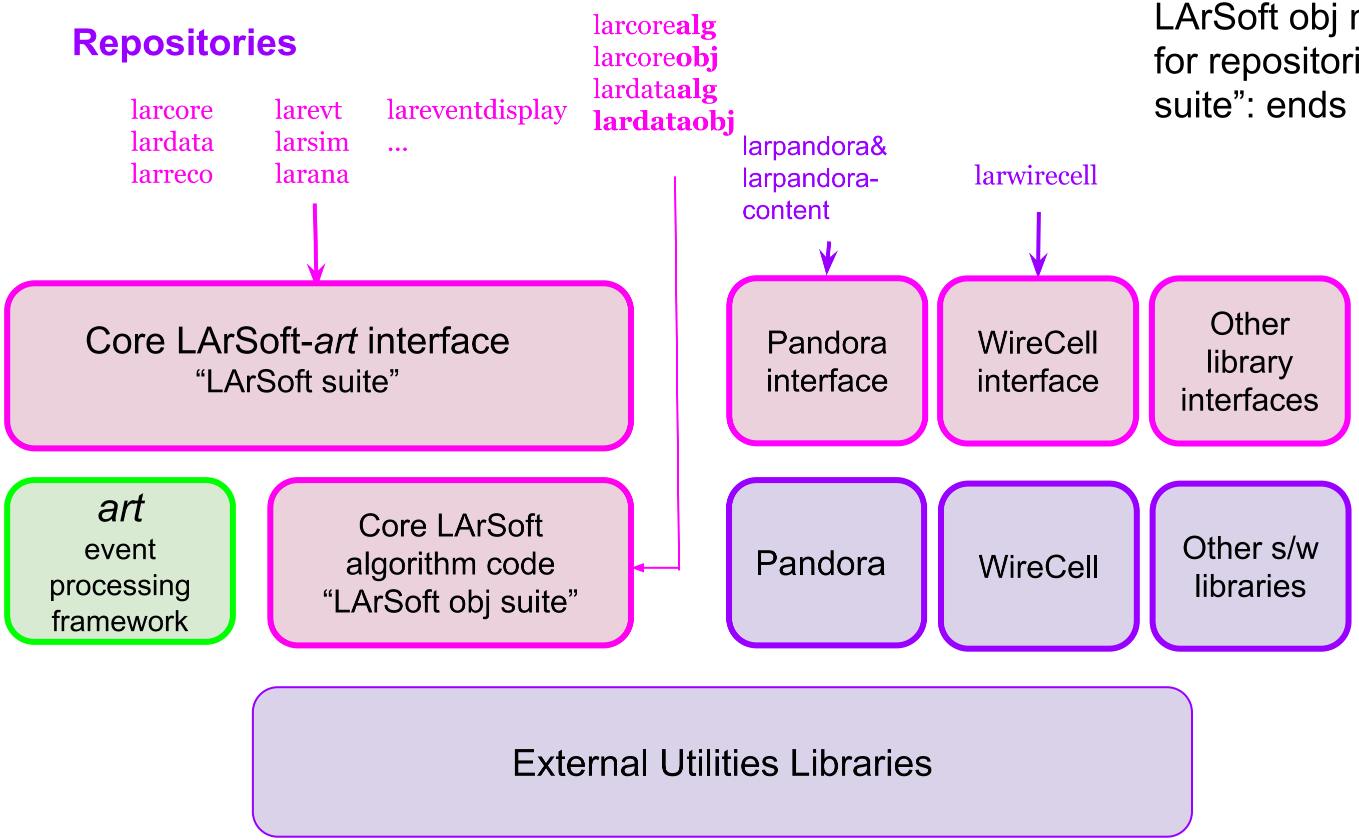
Other s/w libraries

External Utilities Libraries

The smallest build unit is the repository

Structural components of LArSoft

LArSoft obj naming convention for repositories in the "LArSoft suite": ends in "obj" or "alg"



ups and LArSoft

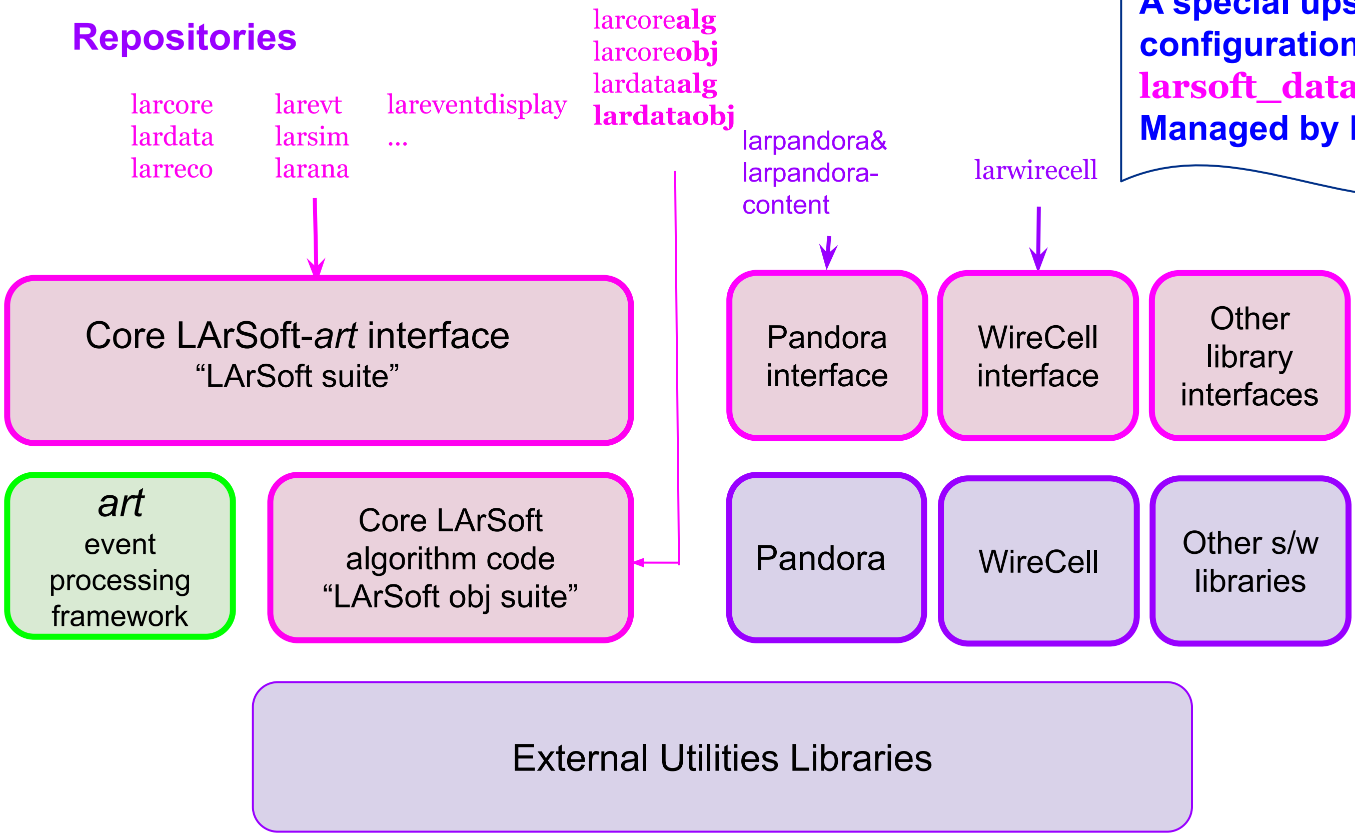
LArSoft is a collection of ups products.

One installed ups product instance per repository.

But not all ups products associated with LArSoft have a repository, such as larsoft_data discussed in the next slide.

ups and LArSoft

A special ups product reserved for larger configuration files (up to a few MB):
larsoft_data
Managed by LArSoft release managers



Repositories

larcore
lardata
larreco

larevt
larsim
larana

lareventdisplay
…

larcore**alg**
larcore**obj**
lardata**alg**
lardataobj

larpandora&
larpandora-content

larwirecell

Core LArSoft-*art* interface
"LArSoft suite"

Pandora interface

WireCell interface

Other library interfaces

art
event processing framework

Core LArSoft algorithm code
"LArSoft obj suite"

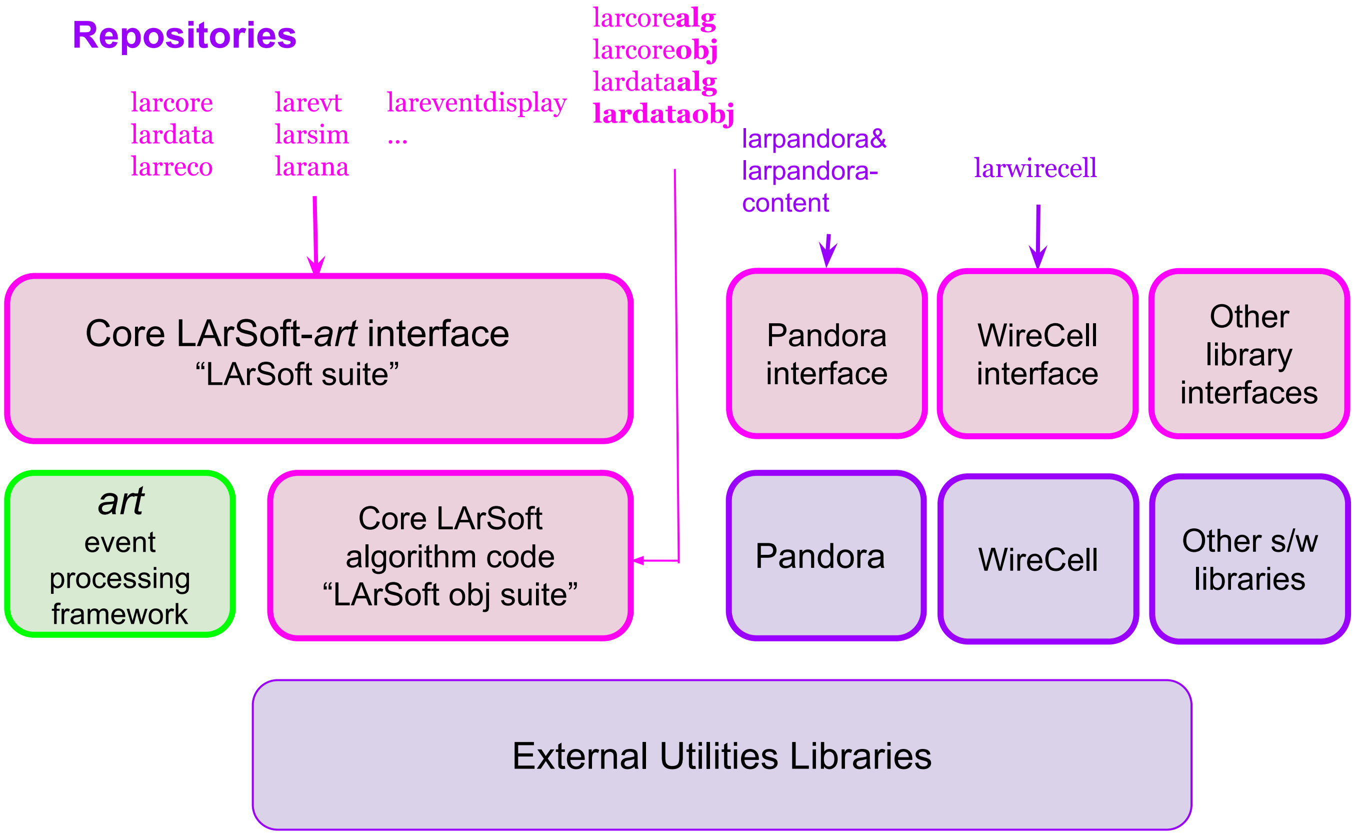
Pandora

WireCell

Other s/w libraries

External Utilities Libraries

ups and LArSoft



Repositories

larcore
lardata
larreco

larevt
larsim
larana

lareventdisplay
...

larcore**alg**
larcore**obj**
lardata**alg**
lardataobj

larpandora&
larpandora-content

larwirecell

Core LArSoft-*art* interface
“LArSoft suite”

Pandora interface

WireCell interface

Other library interfaces

art
event processing framework

Core LArSoft algorithm code
“LArSoft obj suite”

Pandora

WireCell

Other s/w libraries

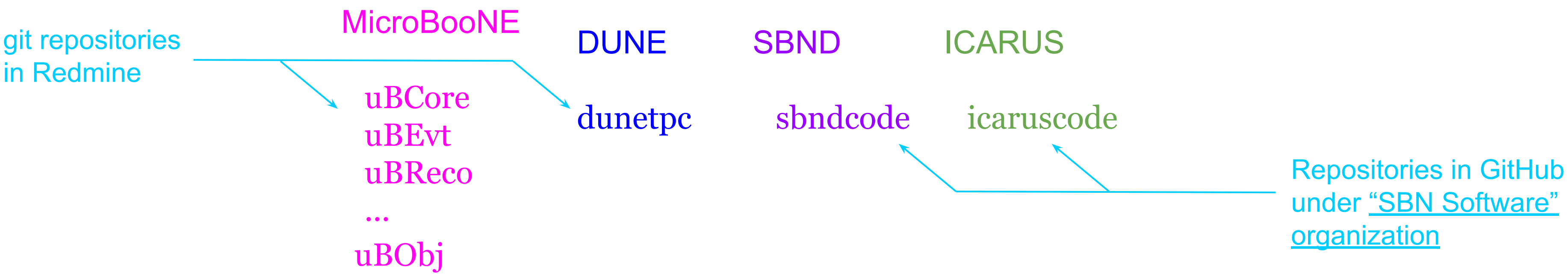
External Utilities Libraries

“larsoft” ups product serves as an umbrella that allows a single setup command for all of LArSoft

larsoft product effectively depends on everything, so “setup larsoft …” sets up everything in a binary-compatible way.

Experiment code

Experiment repositories



MicroBooNE

uBCore
uBEvt
uBReco
...
uBObj

DUNE

dunetpc

SBND

sbndcode

ICARUS

icaruscode

git repositories in Redmine

Repositories in GitHub under "SBN Software" organization

Some experiment code may, strictly speaking, be *art* independent.

Most (all but MicroBooNE) lack required repository structure to build independently of art.

Experiment code

Experiment UPS products

MicroBooNE

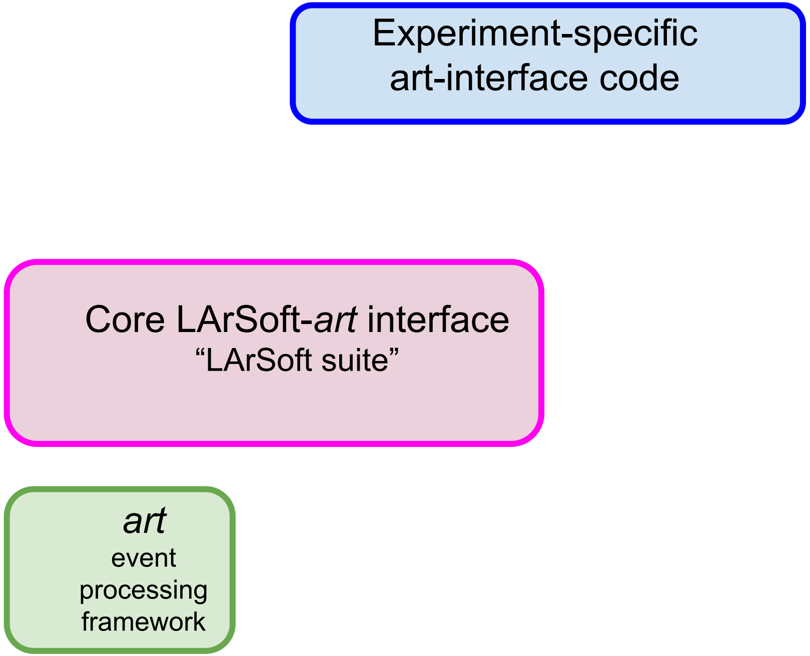
uboonecode (umbrella product)

uBCore
uBEvt
uBReco
…
uBObj

DUNE	SBND	ICARUS
dunetpc	sbndcode	icaruscode

Except for MicroBooNE, umbrella products have the same name as the repositories

Structural components of LArSoft



Experiment-specific art-interface code

Core LArSoft-*art* interface
"LArSoft suite"

art
event processing framework

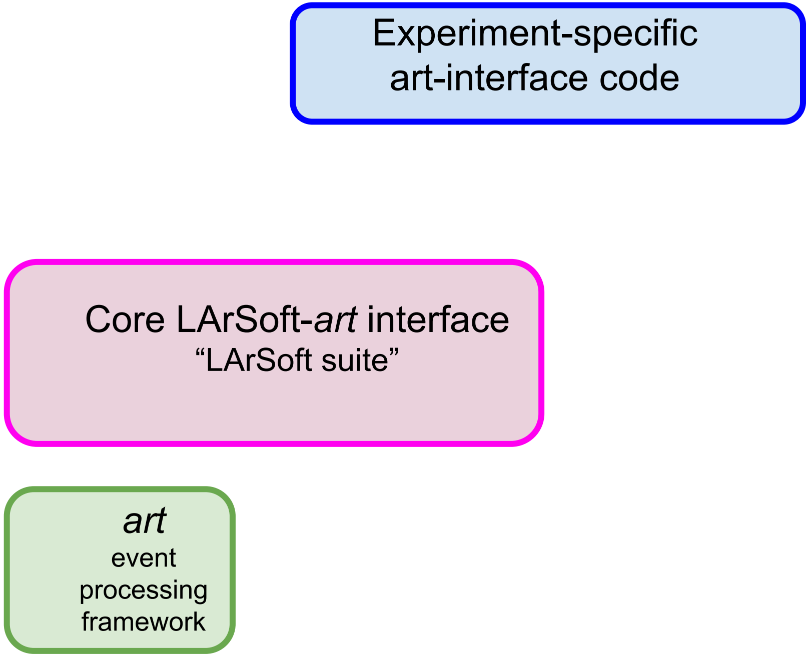
The "*art* interface" code

art module

- art::Event
- art::ServiceHandle<*service*>
- art::Handle<*data product*>
- art::make_tool<*tool type*>
- …

The **event class**, **modules**, **services / service registry**, **handles** (all types), and **associated pre-processor directives**, etc., are all part of *art* interface

Structural components of LArSoft



Experiment-specific art-interface code

Core LArSoft-*art* interface "LArSoft suite"

art event processing framework

The "*art* interface" code

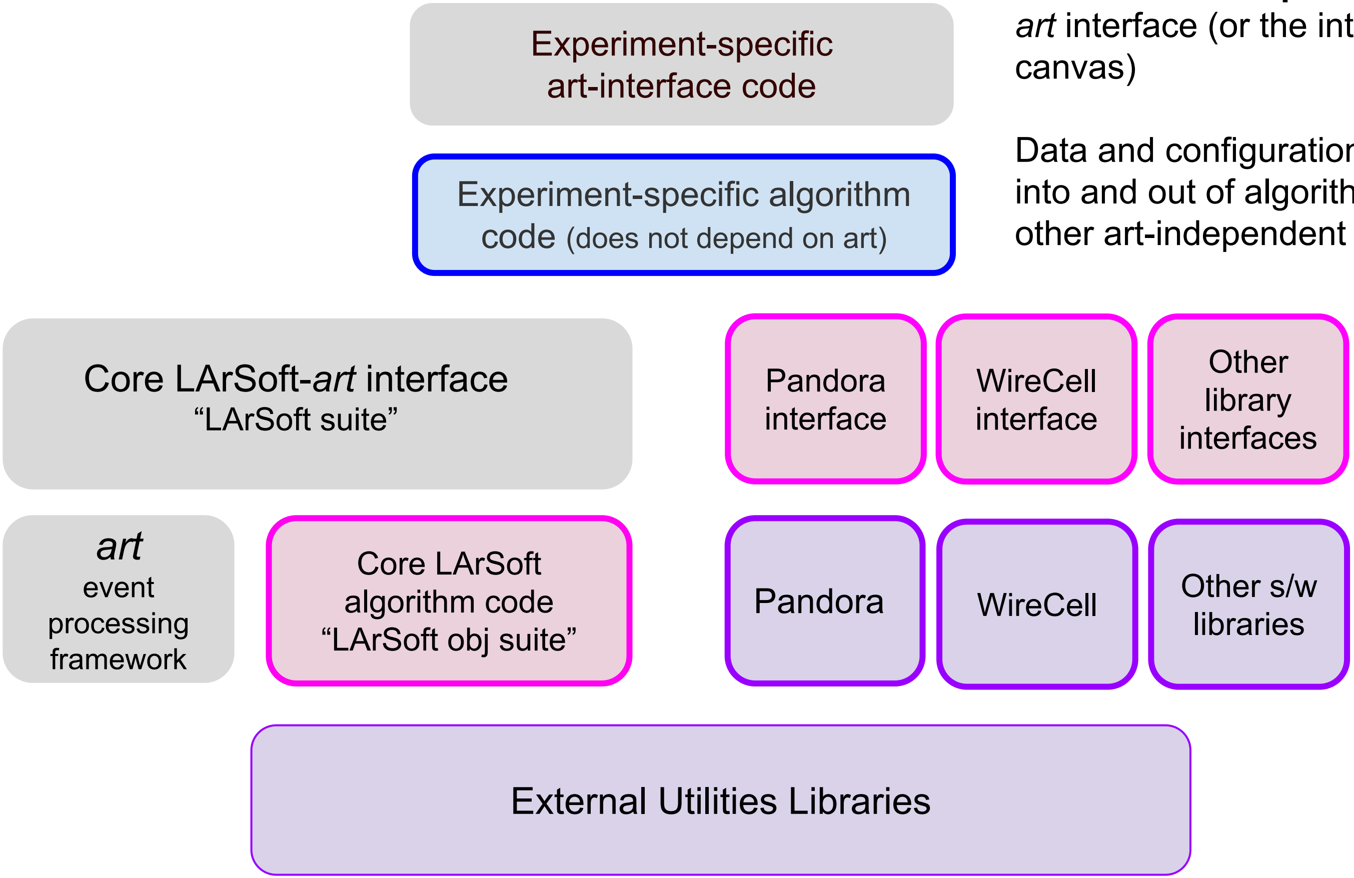
art module

art::Event
art::ServiceHandle<*service*>
art::Handle<*data product*>
art::make_tool<*tool type*>
…

The **event record**, **modules**, **services / service registry**, **handles** (all types), and **associated pre-processor directives**, etc., are all part of *art* interface

Modules should be used to get services, service-providers, parameter sets and data products, and to create tools, which should then be **passed** to algorithm code

art-independent Code



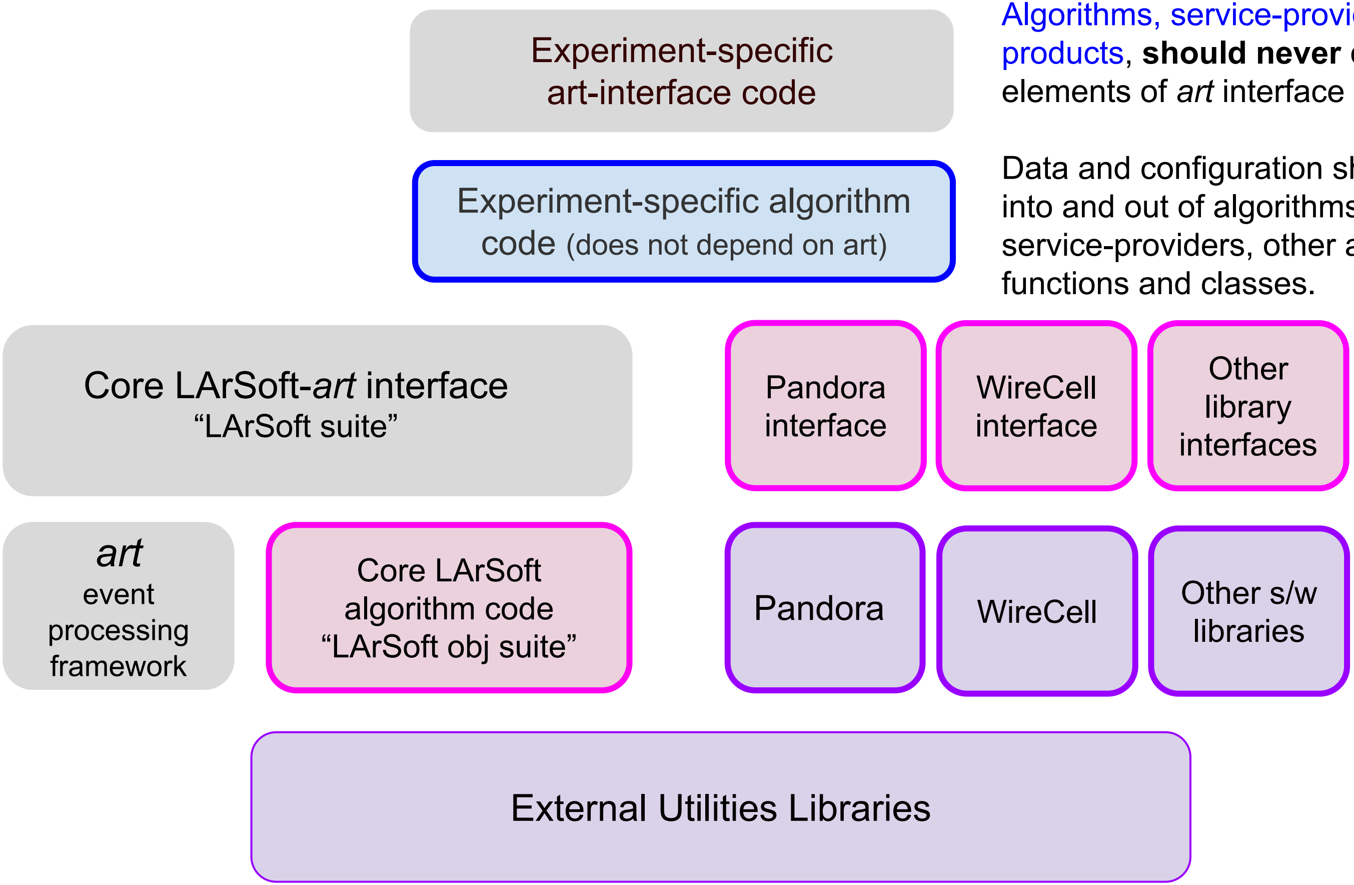
Algorithms, service-providers, data products, **should never depend on any** elements of *art* interface (or the interface provided by canvas)

Data and configuration should be **passed** into and out of algorithms, service-providers, other art-independent functions and classes.

art-independent Code

Algorithms, service-providers, data products, **should never depend on any** elements of *art* interface

Data and configuration should be **passed** into and out of algorithms, service-providers, other art-independent functions and classes.



Experiment-specific art-interface code

Experiment-specific algorithm code (does not depend on art)

Core LArSoft-*art* interface "LArSoft suite"

Pandora interface

WireCell interface

Other library interfaces

art event processing framework

Core LArSoft algorithm code "LArSoft obj suite"

Pandora

WireCell

Other s/w libraries

External Utilities Libraries

Note: fhicl-cpp and message_facility are independent of *art*

- "*art* independent code" **may** include FHiCL parameter sets, message_facility calls, but need not

Why framework independence matters

Code that does not depend on *art* and all the attendant dependencies can:

- Be developed, built in a lightweight stand-alone environment
- Have easily constructed unit tests to check proper functioning
- Be used in alternate event processing / analysis frameworks and contexts
- Be used with *art* gallery
  - Provides lightweight access to art/ROOT files outside of art
  - Widely used both as analysis and development environment
  - The entire LArSoft Obj suite can be used in gallery

  More information at https://art.fnal.gov/gallery/

Code releases and distribution

LArSoft releases

A release contains all LArSoft code, ups products in a frozen state for distribution

Several types of releases

- Production
- Integration
- Test release
- Release candidate

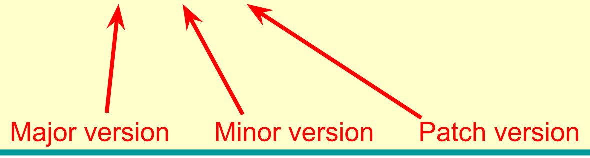
LArSoft releases

A release contains all LArSoft code, ups products in a frozen state for distribution

Several types of releases

- Production
- Integration
- Test release
- Release candidate

- Any release designated as “production” by an experiment
  - Contents approved by the experiment
- Typically used for large-scale processing campaigns
- Created on demand
- Retained indefinitely on disk
- Numbering: vxx_yy_zz, e.g., v08_22_00



Major version

Minor version

Patch version

Details on [“LArSoft release naming and retention policy” wiki page]

LArSoft releases

A release contains all LArSoft code, ups products in a frozen state for distribution

Several types of releases

- Production
- Integration
- Test release
- Release candidate

- Any release designated as “production” by an experiment
  - Contents approved by the experiment
- Typically used for large-scale processing campaigns
- Created on demand
- Retained indefinitely on disk
- Numbering: vxx_yy_zz, e.g., v08_22_00
  - Extend numbering for updates: vxx_yy_zz_aa, e.g., v08_22_00_01, ...

Details on “LArSoft release naming and retention policy” wiki page

LArSoft releases

A release contains all LArSoft code, ups products in a frozen state for distribution

Several types of releases

- Production
- Integration
- Test release
- Release candidate

- Created weekly, or on demand for special purposes
- Provides a stable code base for development that is close to the head of repositories
- Contents approved via pull requests
  - Major changes also require approval at LArSoft Coordination Meetings
- May be removed without notice after about a month (though has never happened…)
- Numbering: vxx_yy_zz (same sequence as production releases)

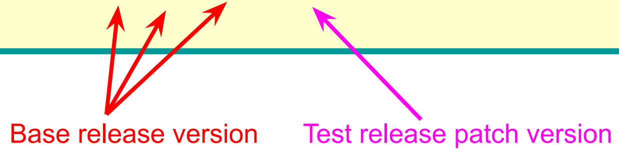
LArSoft releases

A release contains all LArSoft code, ups products in a frozen state for distribution

Several types of releases

- Production
- Integration
- Test release
- Release candidate

- Created to allow experiments to test a new product or new produce version (e.g., Genie, Geant4, art (sometimes)) on top of a known release
- Identical to some base integration or production release except for that product version + any adaptations needed for integration
- Retained on disk until testing is completed
- Numbering: vxx_yy_zz_kk



LArSoft releases

A release contains all LArSoft code, ups products in a frozen state for distribution

Several types of releases

- Production
- Integration
- Test release
- Release candidate

- Created to allow experiments to test a new major version of LArSoft.
  - Sometimes (rarely), a major change to a critical underlying product will trigger this condition
- Retained on disk until testing is completed
- Numbering: vxx_yy_zz_rcn

Target release version

Release candidate version

LArSoft releases

A release contains all LArSoft code, ups products in a frozen state for distribution

Several types of releases

- Production
- Integration
- Test release
- Release candidate

The list of all LArSoft releases, the purpose, significant changes listed on the "LArSoft release list" wiki page (https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/LArSoft_release_list)

Each entry has a link to release notes for that release

LArSoft code distribution

LArSoft releases are distributed via two mechanisms

- cvmfs
  - CERN virtual file system
  - Appears as locally mounted disk area
    - /cvmfs/larsoft.opensciencegrid.org/products/larsoft
- Binary and source tarballs
  - Downloadable from scisoft.fnal.gov
    - https://scisoft.fnal.gov/
  - Instructions for installing, building (when needed) are linked from the release notes

LArSoft code distribution

Every release is distributed in several build variants

- Operating system
- Combination of compiler version + other build flags
- Optimized versus debug versions

Distinguished during setup by

- The current operating system (or as specified in the setup command)
- Qualifiers specified in the setup command

More on this later

Supported platforms

- “Supported platforms”
  - Builds actively supported
  - Code runs and works as intended (as reported by CI system)
  - Source and binary distributions available on cvmfs and scisoft.fnal.gov

  Currently includes:

  - SL7

Supported platforms

- “Known to work”
  - We know of someone (usually us!) who has succeeded in building and running
  - LArSoft does not officially support builds or distribution

A special “best effort” category exists in this space

- Includes operating systems considered as important to LArSoft developer community
- Support on-demand builds, or regular builds after release of “supported platform” distributions
- May or may not include CI system support

Currently includes:

- Ubuntu LTS 20: on-demand, no CI system support

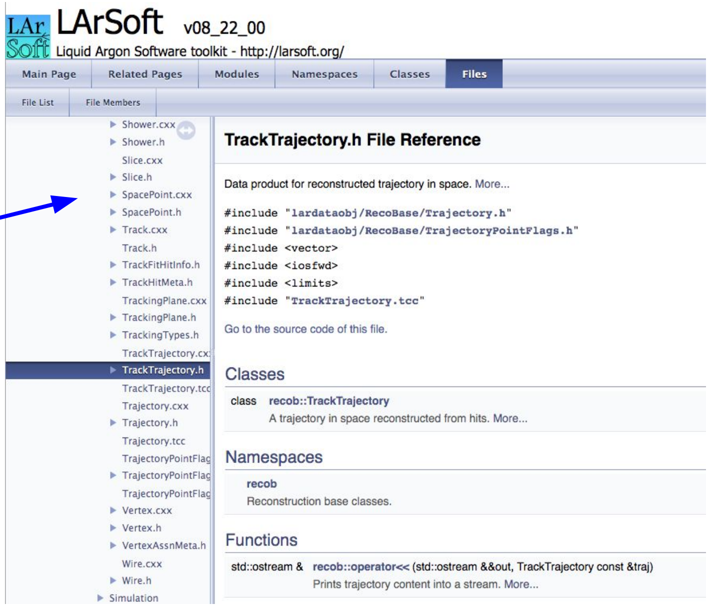
End-user / developer resources

Documentation

Doxygen: http://nusoft.fnal.gov/larsoft/doxsvn/html/

- Auto-generated documentation from markup embedded in source comments

“File” view



TrackTrajectory.h File Reference

Data product for reconstructed trajectory in space. More...

```
#include "lardataobj/RecoBase/Trajectory.h"
#include "lardataobj/RecoBase/TrajectoryPointFlags.h"
#include <vector>
#include <iosfwd>
#include <limits>
#include "TrackTrajectory.tcc"
```

Go to the source code of this file.

Classes

class **recob::TrackTrajectory**
A trajectory in space reconstructed from hits. More...

Namespaces

recob
Reconstruction base classes.

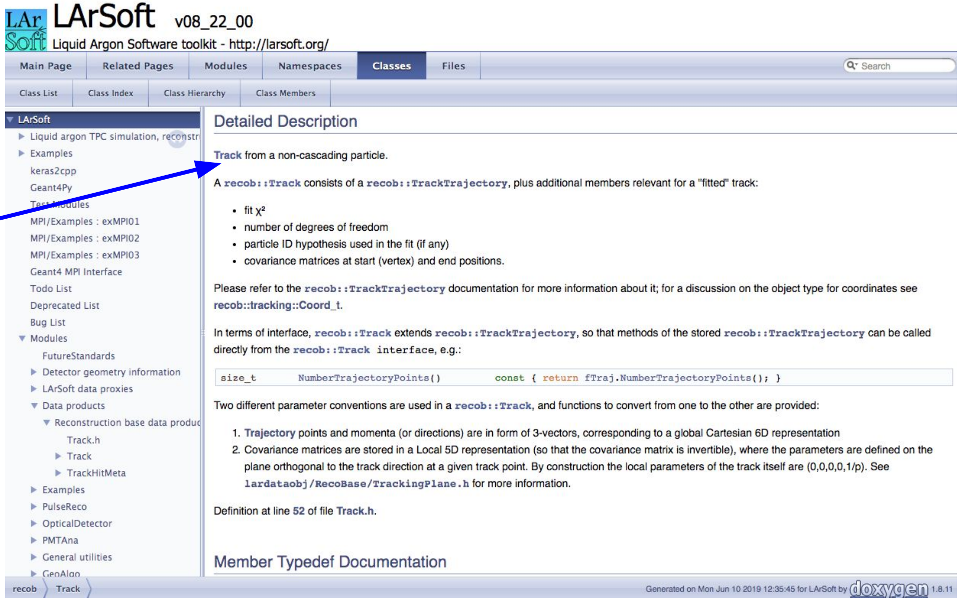
Functions

std::ostream & **recob::operator<<** (std::ostream &&out, TrackTrajectory const &traj)
Prints trajectory content into a stream. More...

Documentation

Doxygen: http://nusoft.fnal.gov/larsoft/doxsvn/html/

- Auto-generated documentation from markup embedded in source comments



Detailed Description

Track from a non-cascading particle.

A `recob::Track` consists of a `recob::TrackTrajectory`, plus additional members relevant for a "fitted" track:

- fit χ^2
- number of degrees of freedom
- particle ID hypothesis used in the fit (if any)
- covariance matrices at start (vertex) and end positions.

Please refer to the `recob::TrackTrajectory` documentation for more information about it; for a discussion on the object type for coordinates see recob::tracking::Coord_t.

In terms of interface, `recob::Track` extends `recob::TrackTrajectory`, so that methods of the stored `recob::TrackTrajectory` can be called directly from the `recob::Track interface`, e.g.:

```
size_t        NumberTrajectoryPoints()           const { return fTraj.NumberTrajectoryPoints(); }
```

Two different parameter conventions are used in a `recob::Track`, and functions to convert from one to the other are provided:

1. Trajectory points and momenta (or directions) are in form of 3-vectors, corresponding to a global Cartesian 6D representation
2. Covariance matrices are stored in a Local 5D representation (so that the covariance matrix is invertible), where the parameters are defined on the plane orthogonal to the track direction at a given track point. By construction the local parameters of the track itself are (0,0,0,0,1/p). See `lardataobj/RecoBase/TrackingPlane.h` for more information.

Definition at line 52 of file Track.h.

Member Typedef Documentation

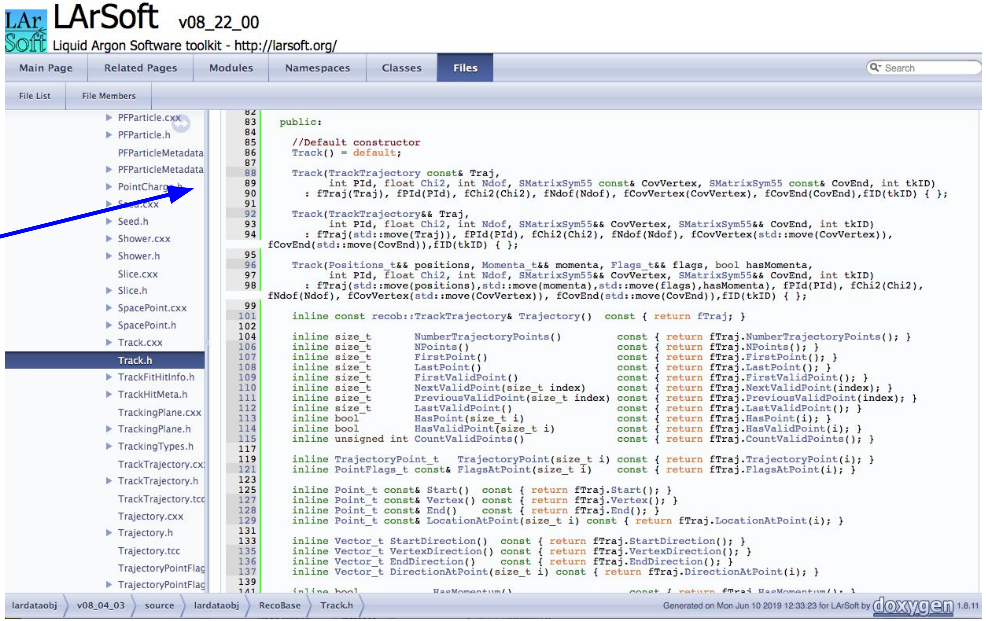
Generated on Mon Jun 10 2019 12:35:45 for LArSoft by doxygen 1.8.11

“Class” view

Documentation

Doxygen: http://nusoft.fnal.gov/larsoft/doxsvn/html/

- Auto-generated documentation from markup embedded in source comments



```
  public:

    //Default constructor
    Track() = default;

    Track(TrackTrajectory const& Traj,
          int PId, float Chi2, int Ndof, SMatrixSym55 const& CovVertex, SMatrixSym55 const& CovEnd, int tkID)
      : fTraj(Traj), fPId(PId), fChi2(Chi2), fNdof(Ndof), fCovVertex(CovVertex), fCovEnd(CovEnd),fID(tkID) { };

    Track(TrackTrajectory&& Traj,
          int PId, float Chi2, int Ndof, SMatrixSym55&& CovVertex, SMatrixSym55&& CovEnd, int tkID)
      : fTraj(std::move(Traj)), fPId(PId), fChi2(Chi2), fNdof(Ndof), fCovVertex(std::move(CovVertex)),
fCovEnd(std::move(CovEnd)),fID(tkID) { };

    Track(Positions_t&& positions, Momenta_t&& momenta, Flags_t&& flags, bool hasMomenta,
          int PId, float Chi2, int Ndof, SMatrixSym55&& CovVertex, SMatrixSym55&& CovEnd, int tkID)
      : fTraj(std::move(positions),std::move(momenta),std::move(flags),hasMomenta), fPId(PId), fChi2(Chi2),
fNdof(Ndof), fCovVertex(std::move(CovVertex)), fCovEnd(std::move(CovEnd)),fID(tkID) { };

    inline const recob::TrackTrajectory& Trajectory()  const { return fTraj; }

    inline size_t       NumberTrajectoryPoints()          const { return fTraj.NumberTrajectoryPoints(); }
    inline size_t       NPoints()                         const { return fTraj.NPoints(); }
    inline size_t       FirstPoint()                      const { return fTraj.FirstPoint(); }
    inline size_t       LastPoint()                       const { return fTraj.LastPoint(); }
    inline size_t       FirstValidPoint()                 const { return fTraj.FirstValidPoint(); }
    inline size_t       NextValidPoint(size_t index)      const { return fTraj.NextValidPoint(index); }
    inline size_t       PreviousValidPoint(size_t index) const { return fTraj.PreviousValidPoint(index); }
    inline size_t       LastValidPoint()                  const { return fTraj.LastValidPoint(); }
    inline bool         HasPoint(size_t i)                const { return fTraj.HasPoint(i); }
    inline bool         HasValidPoint(size_t i)           const { return fTraj.HasValidPoint(i); }
    inline unsigned int CountValidPoints()                const { return fTraj.CountValidPoints(); }

    inline TrajectoryPoint_t   TrajectoryPoint(size_t i) const { return fTraj.TrajectoryPoint(i); }
    inline PointFlags_t const& FlagsAtPoint(size_t i)    const { return fTraj.FlagsAtPoint(i); }

    inline Point_t const& Start()  const { return fTraj.Start(); }
    inline Point_t const& Vertex() const { return fTraj.Vertex(); }
    inline Point_t const& End()    const { return fTraj.End(); }
    inline Point_t const& LocationAtPoint(size_t i) const { return fTraj.LocationAtPoint(i); }

    inline Vector_t StartDirection()  const { return fTraj.StartDirection(); }
    inline Vector_t VertexDirection() const { return fTraj.VertexDirection(); }
    inline Vector_t EndDirection()    const { return fTraj.EndDirection(); }
    inline Vector_t DirectionAtPoint(size_t i) const { return fTraj.DirectionAtPoint(i); }

    inline bool           HasMomentum()                   const { return fTraj.HasMomentum(); }
```

“Source” view

Documentation

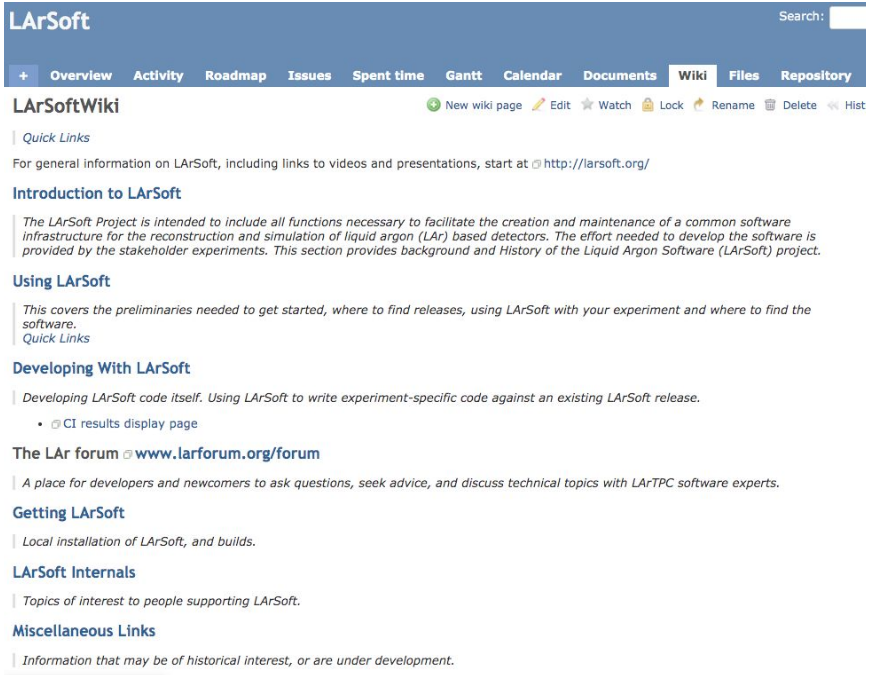
Doxygen: http://nusoft.fnal.gov/larsoft/doxsvn/html/

- Auto-generated documentation from markup embedded in source comments
- Pros:
  - A significant fraction of code includes such comments
  - Should always be up to date with the code you are viewing
- Cons:
  - Provides no high-level view or context
  - Quality varies greatly due to absence of enforceable standards or conventions

LArSoft Redmine site

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

- Technical reference
- Issue tracker



LArSoftWiki

Quick Links

For general information on LArSoft, including links to videos and presentations, start at http://larsoft.org/

Introduction to LArSoft

> *The LArSoft Project is intended to include all functions necessary to facilitate the creation and maintenance of a common software infrastructure for the reconstruction and simulation of liquid argon (LAr) based detectors. The effort needed to develop the software is provided by the stakeholder experiments. This section provides background and History of the Liquid Argon Software (LArSoft) project.*

Using LArSoft

> *This covers the preliminaries needed to get started, where to find releases, using LArSoft with your experiment and where to find the software.*
> *Quick Links*

Developing With LArSoft

> *Developing LArSoft code itself. Using LArSoft to write experiment-specific code against an existing LArSoft release.*

- CI results display page

The LAr forum www.larforum.org/forum

> *A place for developers and newcomers to ask questions, seek advice, and discuss technical topics with LArTPC software experts.*

Getting LArSoft

> *Local installation of LArSoft, and builds.*

LArSoft Internals

> *Topics of interest to people supporting LArSoft.*

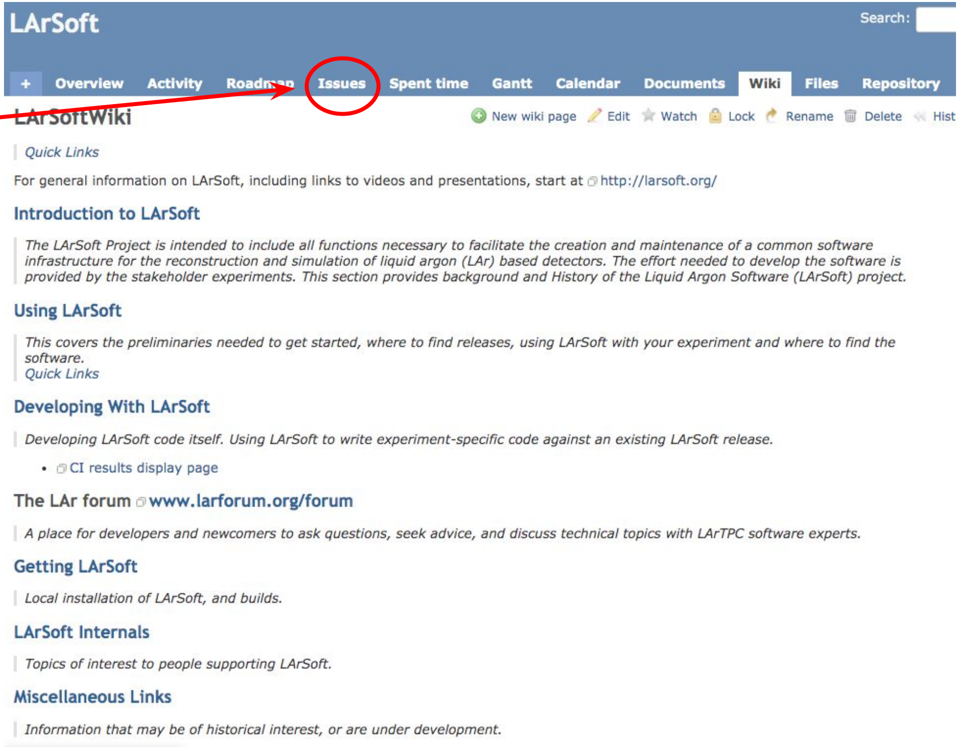
Miscellaneous Links

> *Information that may be of historical interest, or are under development.*

LArSoft Redmine site

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

- Technical reference
- Issue tracker



LArSoftWiki

Quick Links

For general information on LArSoft, including links to videos and presentations, start at http://larsoft.org/

Introduction to LArSoft

> *The LArSoft Project is intended to include all functions necessary to facilitate the creation and maintenance of a common software infrastructure for the reconstruction and simulation of liquid argon (LAr) based detectors. The effort needed to develop the software is provided by the stakeholder experiments. This section provides background and History of the Liquid Argon Software (LArSoft) project.*

Using LArSoft

> *This covers the preliminaries needed to get started, where to find releases, using LArSoft with your experiment and where to find the software.*
> *Quick Links*

Developing With LArSoft

> *Developing LArSoft code itself. Using LArSoft to write experiment-specific code against an existing LArSoft release.*

- CI results display page

The LAr forum www.larforum.org/forum

> *A place for developers and newcomers to ask questions, seek advice, and discuss technical topics with LArTPC software experts.*

Getting LArSoft

> *Local installation of LArSoft, and builds.*

LArSoft Internals

> *Topics of interest to people supporting LArSoft.*

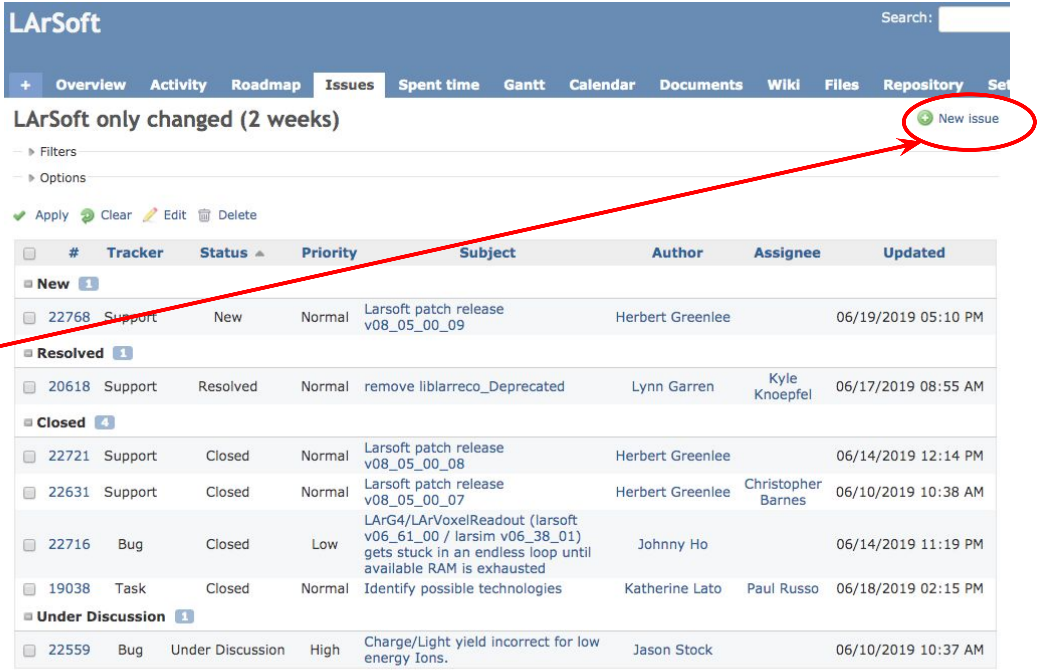
Miscellaneous Links

> *Information that may be of historical interest, or are under development.*

LArSoft Redmine site

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

- Technical reference
- Issue tracker



LArSoft
Search:
+ Overview Activity Roadmap Issues Spent time Gantt Calendar Documents Wiki Files Repository Set
New issue
Filters
Options
Apply Clear Edit Delete

LArSoft only changed (2 weeks)

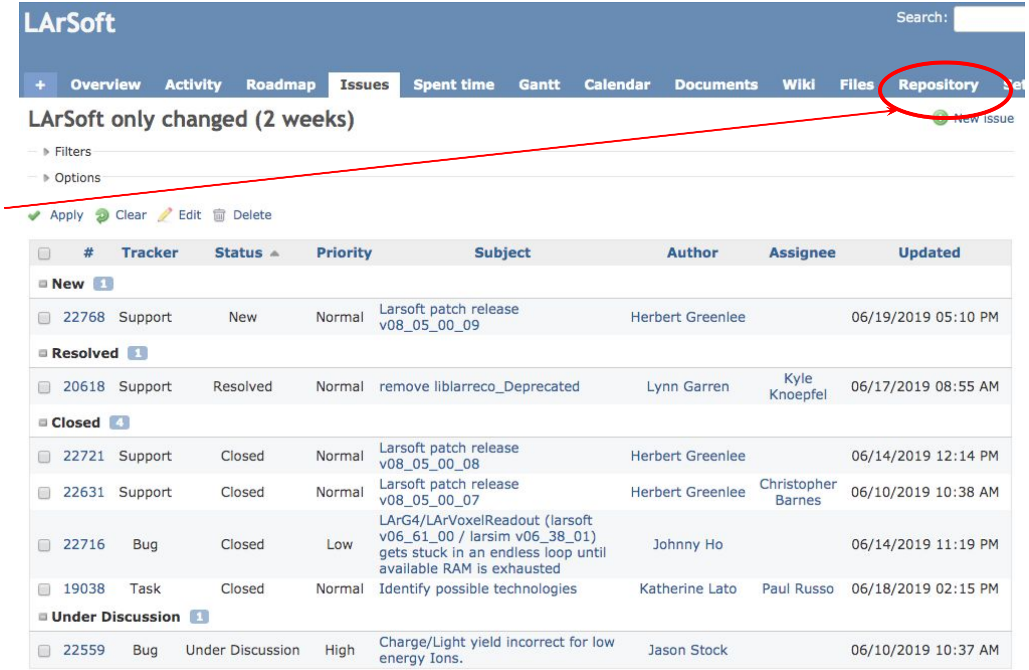
□	#	Tracker	Status ▲	Priority	Subject	Author	Assignee	Updated
New 1								
□	22768	Support	New	Normal	Larsoft patch release v08_05_00_09	Herbert Greenlee		06/19/2019 05:10 PM
Resolved 1								
□	20618	Support	Resolved	Normal	remove liblarreco_Deprecated	Lynn Garren	Kyle Knoepfel	06/17/2019 08:55 AM
Closed 4								
□	22721	Support	Closed	Normal	Larsoft patch release v08_05_00_08	Herbert Greenlee		06/14/2019 12:14 PM
□	22631	Support	Closed	Normal	Larsoft patch release v08_05_00_07	Herbert Greenlee	Christopher Barnes	06/10/2019 10:38 AM
□	22716	Bug	Closed	Low	LArG4/LArVoxelReadout (larsoft v06_61_00 / larsim v06_38_01) gets stuck in an endless loop until available RAM is exhausted	Johnny Ho		06/14/2019 11:19 PM
□	19038	Task	Closed	Normal	Identify possible technologies	Katherine Lato	Paul Russo	06/18/2019 02:15 PM
Under Discussion 1								
□	22559	Bug	Under Discussion	High	Charge/Light yield incorrect for low energy Ions.	Jason Stock		06/10/2019 10:37 AM

Report problems
Make requests
Ask questions
Make suggestions

LArSoft Redmine site

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

- Technical reference
- Issue tracker
- “Repositories” no longer used
  - – Everything is in GitHub



LArSoft only changed (2 weeks)

□	#	Tracker	Status ▲	Priority	Subject	Author	Assignee	Updated
New 1								
□	22768	Support	New	Normal	Larsoft patch release v08_05_00_09	Herbert Greenlee		06/19/2019 05:10 PM
Resolved 1								
□	20618	Support	Resolved	Normal	remove liblarreco_Deprecated	Lynn Garren	Kyle Knoepfel	06/17/2019 08:55 AM
Closed 4								
□	22721	Support	Closed	Normal	Larsoft patch release v08_05_00_08	Herbert Greenlee		06/14/2019 12:14 PM
□	22631	Support	Closed	Normal	Larsoft patch release v08_05_00_07	Herbert Greenlee	Christopher Barnes	06/10/2019 10:38 AM
□	22716	Bug	Closed	Low	LArG4/LArVoxelReadout (larsoft v06_61_00 / larsim v06_38_01) gets stuck in an endless loop until available RAM is exhausted	Johnny Ho		06/14/2019 11:19 PM
□	19038	Task	Closed	Normal	Identify possible technologies	Katherine Lato	Paul Russo	06/18/2019 02:15 PM
Under Discussion 1								
□	22559	Bug	Under Discussion	High	Charge/Light yield incorrect for low energy Ions.	Jason Stock		06/10/2019 10:37 AM

LArSoft GitHub

https://github.com/LArSoft

LArSoft

Software for Liquid Argon time projection chambers

Fermi National Accelerator Laborat...

http://larsoft.org

LArSoft GitHub

https://github.com/LArSoft

Working with GitHub - https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/Working_with_Github

In order to develop and contribute LArSoft code, you will need to have a personal GitHub account.

- If you don't have one already, go to: https://github.com/join
  - Follow the instructions to create a new account. Make sure you either use a username that people will easily recognize, or specify your real name, so that people know who issued the pull request.
- If you have an account, use the "Sign in" dialog at https://github.com/login

Contributed code uses the pull request feature.

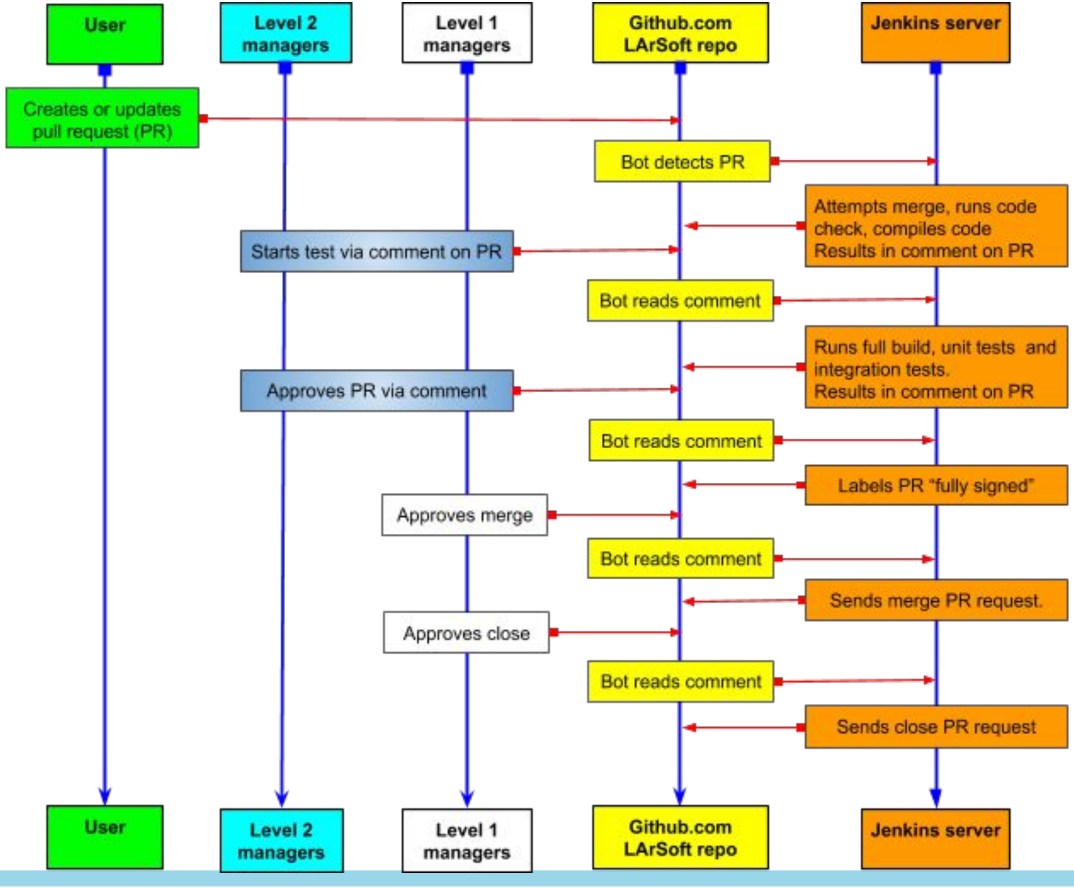
LArSoft GitHub

Developers must initiate a pull-request for the specific change to be merged, since most users will not have privilege to commit directly to the LArSoft repositories on GitHub. In order to create a pull request, a person must first:

- Have a properly configured personal GitHub account
- Push the feature branch to the forked LArSoft repositories in their personal GitHub account

Creating the pull request then triggers the workflow shown on the next page.

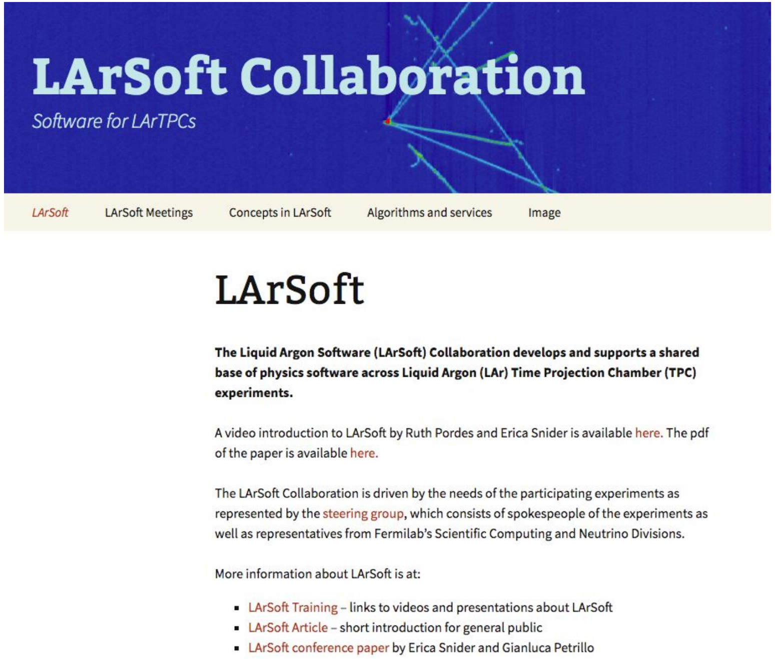
LArSoft GitHub - Overview of the pull request testing and approval workflow



LArSoft.org

https://larsoft.org/

- Organizational information about the collaboration
  - Governance structure
  - Meeting notes
- High-level documentation
- Links to training information / sessions



LArSoft

The Liquid Argon Software (LArSoft) Collaboration develops and supports a shared base of physics software across Liquid Argon (LAr) Time Projection Chamber (TPC) experiments.

A video introduction to LArSoft by Ruth Pordes and Erica Snider is available here. The pdf of the paper is available here.

The LArSoft Collaboration is driven by the needs of the participating experiments as represented by the steering group, which consists of spokespeople of the experiments as well as representatives from Fermilab's Scientific Computing and Neutrino Divisions.

More information about LArSoft is at:

- LArSoft Training – links to videos and presentations about LArSoft
- LArSoft Article – short introduction for general public
- LArSoft conference paper by Erica Snider and Gianluca Petrillo

LArSoft CI system

Documentation: https://cdcvs.fnal.gov/redmine/projects/lar_ci/wiki

Monitoring app: http://lar-ci-history.fnal.gov/LarCI/app

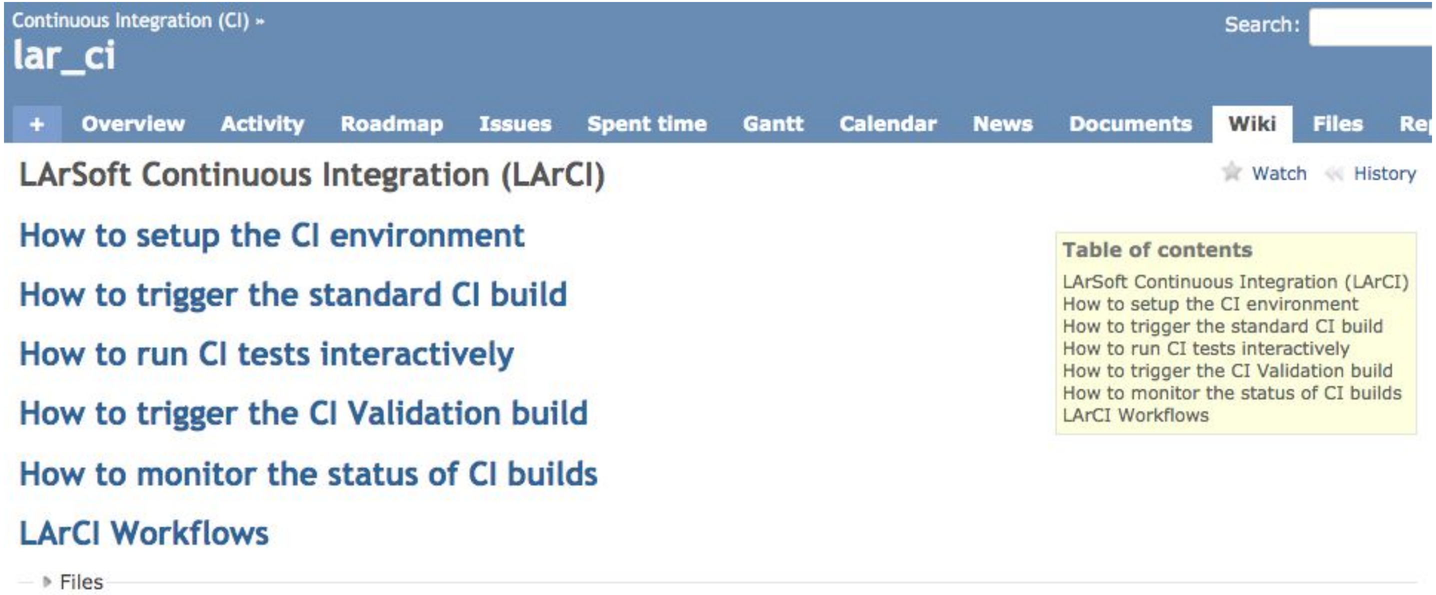
- Drives both rapid turn-around CI testing and more comprehensive validation workflows and testing
- Users can run tests locally prior to committing code, or launch jobs to look at specified combinations of branches

LArSoft CI system

Documentation: https://cdcvs.fnal.gov/redmine/projects/lar_ci/wiki

Monitoring app: http://lar-ci-history.fnal.gov/LarCI/app

lar_ci wiki page



LArSoft Continuous Integration (LArCI)

How to setup the CI environment

How to trigger the standard CI build

How to run CI tests interactively

How to trigger the CI Validation build

How to monitor the status of CI builds

LArCI Workflows

Table of contents

▸ Files

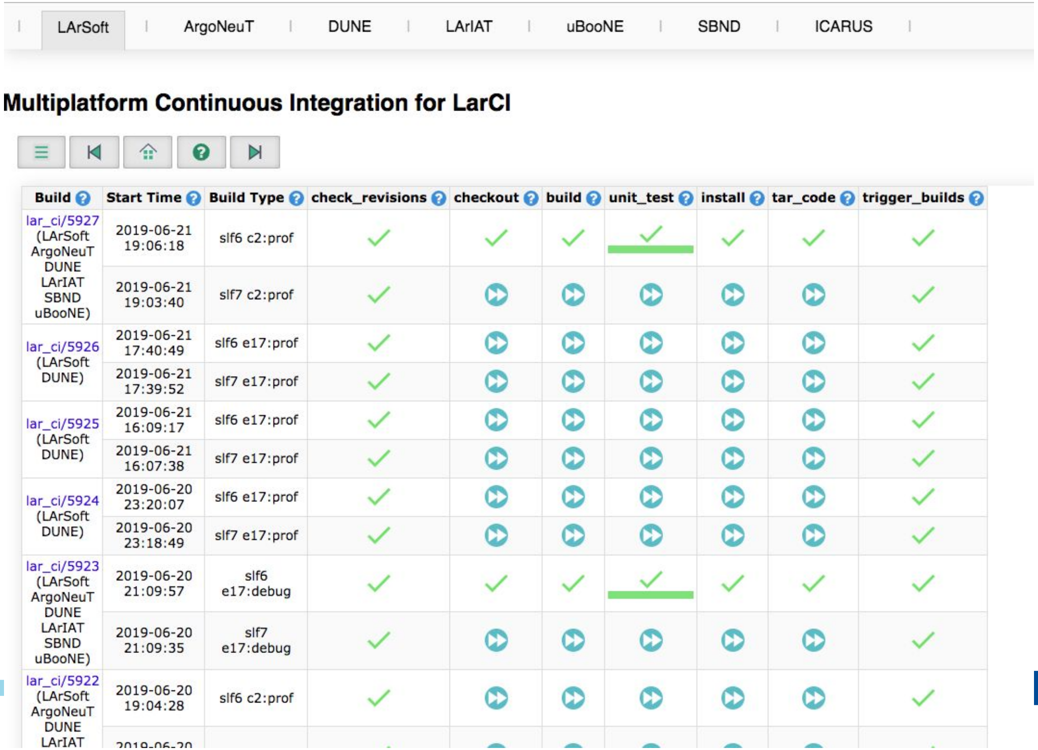
LArSoft CI system

Documentation: https://cdcvs.fnal.gov/redmine/projects/lar_ci/wiki

Monitoring app: http://lar-ci-history.fnal.gov/LarCI/app

Monitoring app

Drill-down by experiment to see test results at increasingly fine detail



Multiplatform Continuous Integration for LarCI

Build	Start Time	Build Type	check_revisions	checkout	build	unit_test	install	tar_code	trigger_builds
lar_ci/5927 (LArSoft ArgoNeuT DUNE LArIAT SBND uBooNE)	2019-06-21 19:06:18	slf6 c2:prof	✓	✓	✓	✓	✓	✓	✓
	2019-06-21 19:03:40	slf7 c2:prof	✓	⏩	⏩	⏩	⏩	⏩	✓
lar_ci/5926 (LArSoft DUNE)	2019-06-21 17:40:49	slf6 e17:prof	✓	⏩	⏩	⏩	⏩	⏩	✓
	2019-06-21 17:39:52	slf7 e17:prof	✓	⏩	⏩	⏩	⏩	⏩	✓
lar_ci/5925 (LArSoft DUNE)	2019-06-21 16:09:17	slf6 e17:prof	✓	⏩	⏩	⏩	⏩	⏩	✓
	2019-06-21 16:07:38	slf7 e17:prof	✓	⏩	⏩	⏩	⏩	⏩	✓
lar_ci/5924 (LArSoft DUNE)	2019-06-20 23:20:07	slf6 e17:prof	✓	⏩	⏩	⏩	⏩	⏩	✓
	2019-06-20 23:18:49	slf7 e17:prof	✓	⏩	⏩	⏩	⏩	⏩	✓
lar_ci/5923 (LArSoft ArgoNeuT DUNE LArIAT SBND uBooNE)	2019-06-20 21:09:57	slf6 e17:debug	✓	✓	✓	✓	✓	✓	✓
	2019-06-20 21:09:35	slf7 e17:debug	✓	⏩	⏩	⏩	⏩	⏩	✓
lar_ci/5922 (LArSoft ArgoNeuT DUNE LArIAT	2019-06-20 19:04:28	slf6 c2:prof	✓	⏩	⏩	⏩	⏩	⏩	✓
	2019-06-20								

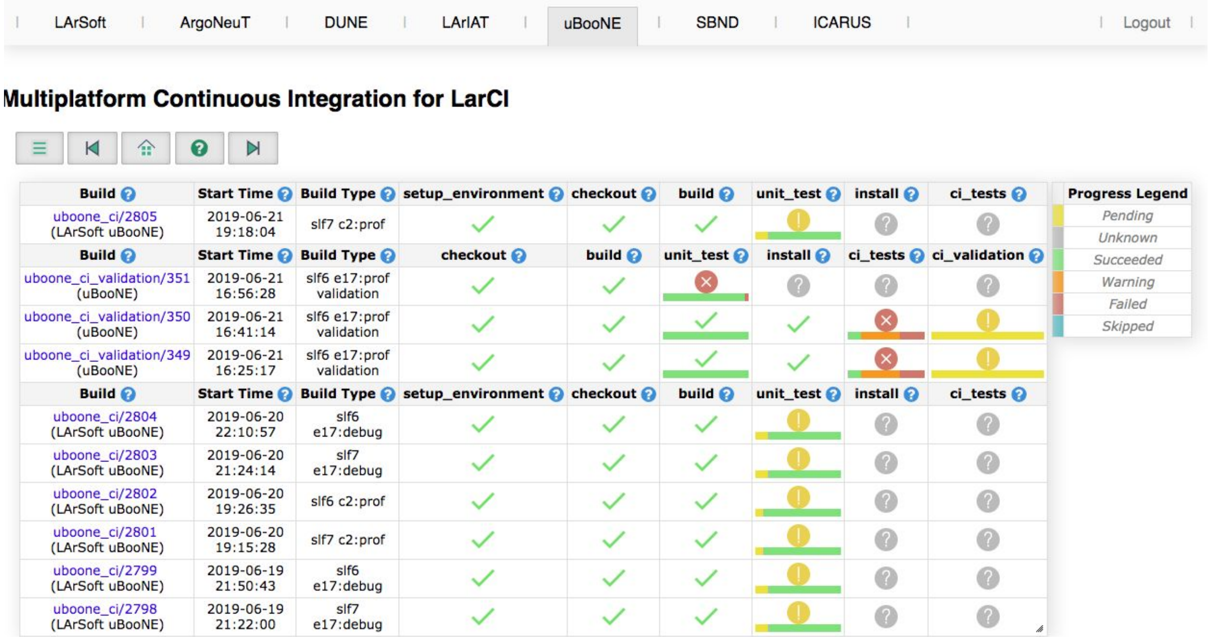
LArSoft CI system

Documentation: https://cdcvs.fnal.gov/redmine/projects/lar_ci/wiki

Monitoring app: http://lar-ci-history.fnal.gov/LarCI/app

Monitoring app

Drill-down by experiment to see test results at increasingly fine detail



Multiplatform Continuous Integration for LarCI

Build	Start Time	Build Type	setup_environment	checkout	build	unit_test	install	ci_tests
uboone_ci/2805 (LArSoft uBooNE)	2019-06-21 19:18:04	slf7 c2:prof	✓	✓	✓	!	?	?
Build	**Start Time**	**Build Type**	**checkout**	**build**	**unit_test**	**install**	**ci_tests**	**ci_validation**
uboone_ci_validation/351 (uBooNE)	2019-06-21 16:56:28	slf6 e17:prof validation	✓	✓	✗	?	?	?
uboone_ci_validation/350 (uBooNE)	2019-06-21 16:41:14	slf6 e17:prof validation	✓	✓	✓	✓	✗	!
uboone_ci_validation/349 (uBooNE)	2019-06-21 16:25:17	slf6 e17:prof validation	✓	✓	✓	✓	✗	!
Build	**Start Time**	**Build Type**	**setup_environment**	**checkout**	**build**	**unit_test**	**install**	**ci_tests**
uboone_ci/2804 (LArSoft uBooNE)	2019-06-20 22:10:57	slf6 e17:debug	✓	✓	✓	!	?	?
uboone_ci/2803 (LArSoft uBooNE)	2019-06-20 21:24:14	slf7 e17:debug	✓	✓	✓	!	?	?
uboone_ci/2802 (LArSoft uBooNE)	2019-06-20 19:26:35	slf6 c2:prof	✓	✓	✓	!	?	?
uboone_ci/2801 (LArSoft uBooNE)	2019-06-20 19:15:28	slf7 c2:prof	✓	✓	✓	!	?	?
uboone_ci/2799 (LArSoft uBooNE)	2019-06-19 21:50:43	slf6 e17:debug	✓	✓	✓	!	?	?
uboone_ci/2798 (LArSoft uBooNE)	2019-06-19 21:22:00	slf7 e17:debug	✓	✓	✓	!	?	?

	Progress Legend
🟨	*Pending*
⬜	*Unknown*
🟩	*Succeeded*
🟧	*Warning*
🟥	*Failed*
🟦	*Skipped*

SciSoft support team

Provides support for LArSoft (among many other software projects, e.g., *art*) via:

- User support
- Technical expertise, problem solving
- Software solutions
- Architecture maintenance and development
- LArSoft work plan execution
- Release management
- Project management

SciSoft support team

Team members:

- Developers / experts / user support
  - Vito di Benedetto
  - Patrick Gartung
  - Chris Green
  - Robert Hatcher
  - Saba Sehrish
  - Mike Wang
- Leaders
  - Kyle Knoepfel
  - Erica Snider
- LArSoft project technical lead
  - Erica Snider

Email to scisoft-team@fnal.gov

The end