LArSoft: toolkit for simulation, reconstruction and analysis of liquid argon TPC detectors

presented by
Chris Jones, *Fermilab*
in lieu of:

Erica Snider
LArSoft Project Lead
Fermilab

Gianluca Petrillo
Fermilab

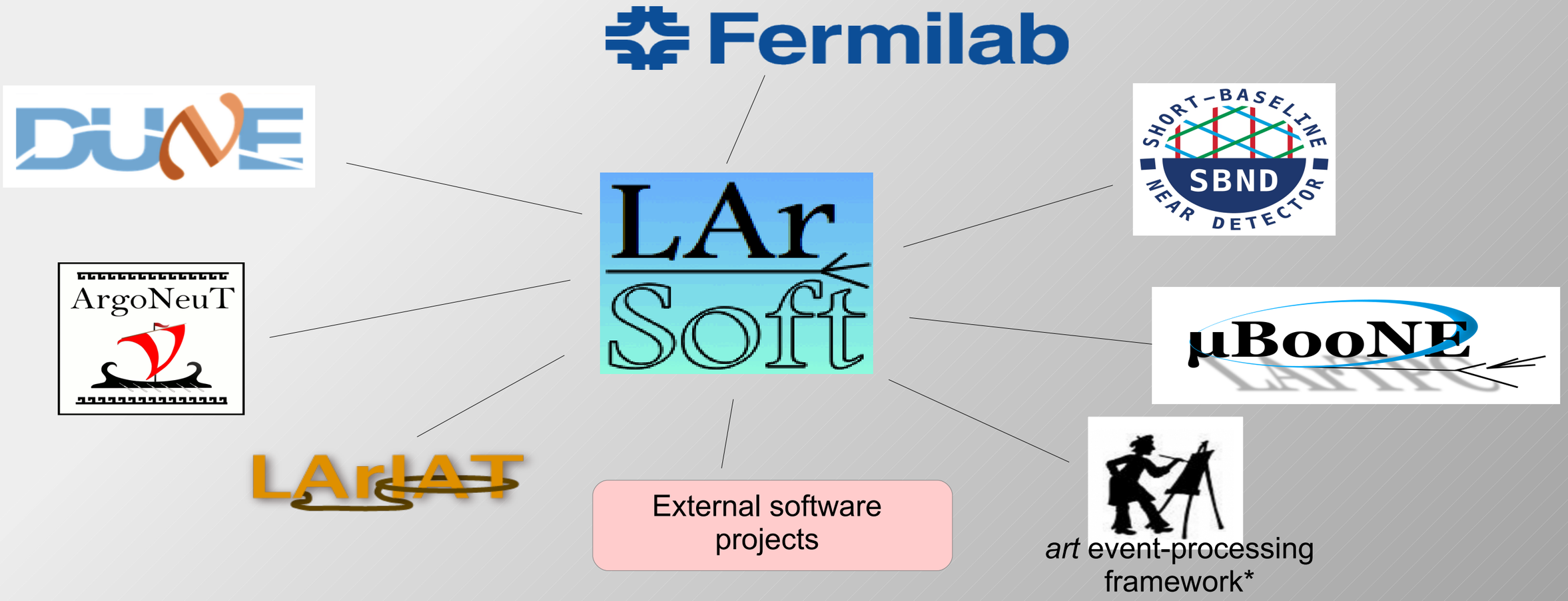
on behalf of the LArSoft project

CHEP 2016
October 8-14, 2016
San Francisco, CA

Outline

- What is LArSoft?
- Design principles and practices
- The development environment
- Coordination of collaboration work
- Future plans
- Summary

What is LArSoft? (1) **A collaboration** of experiments, labs, university groups and software projects

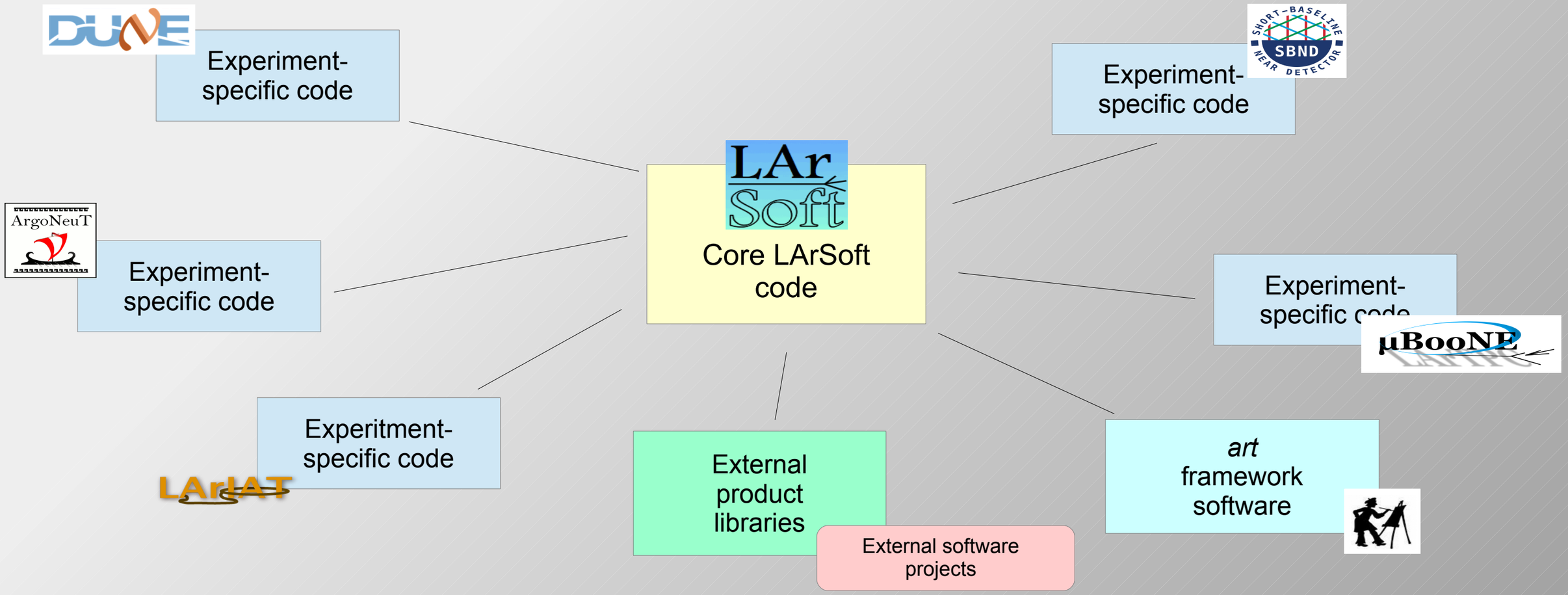


Goal: To provide integrated, experiment-independent software tools for the simulation, reconstruction and analysis for liquid argon (LAr) TPC neutrino experiments

* http://art.fnal.gov/

What is LArSoft?

(2) **The shared body of code** produced by and for the collaboration that performs those tasks, and that interfaces with experiment-specific, *art* framework and external product software



Experiments, software projects **contribute to and use** common “core” LArSoft code

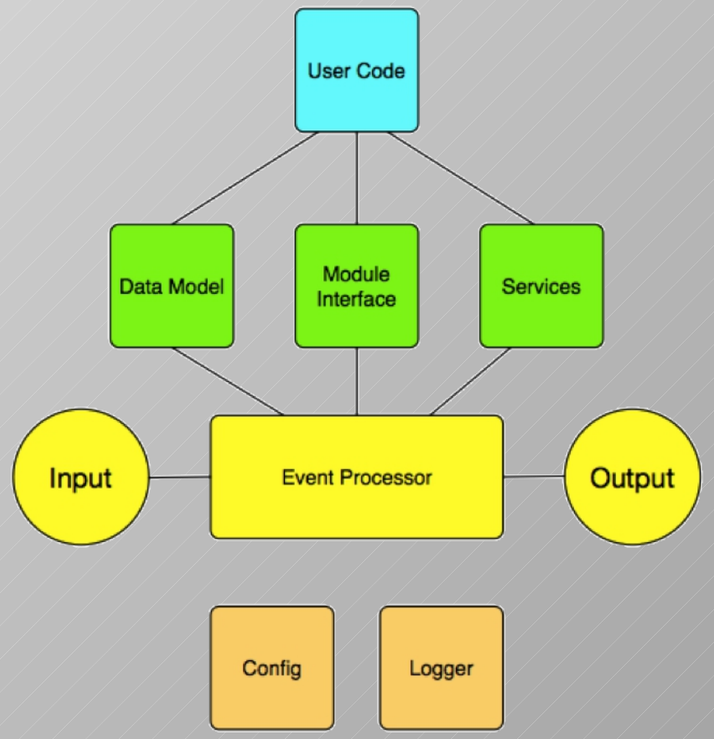
230k lines of C++ in core LArSoft
450k+ lines including experiment code
110 authors from more than 25 institutions

The LArSoft Project

- Organized by and for the collaboration
- The means by which **expertise and software is shared across experiments**
- A core support team: provisioning and support for the core framework, architecture, design, release management, testing and roadmap activities across experiments
- Facilitates adding value through collaboration, including:
  - Increasing the quality and effectiveness of algorithm code
  - Providing clean integration with other products
  - Supporting new ideas and proposals that build out from existing capabilities
  - Realizing reductions in the total effort needed across experiments
- One Fermilab's centralized activities towards common software and computing services across experiments

The *art* event-processing framework

- LArSoft is built on the *art* event-processing framework *
  - Provides facilities to:
    - Define experiment-written “modules” that perform steps in a workflow
    - Define common resources or “services” to all modules
    - Configure the execution of these modules and services
    - Handle experiment-defined data structures (“data products”)
    - Read and write files containing these data products
    - Track the provenance of the data generated during execution
  - *art* is used by and supported for most Fermilab-based experiments



* http://art.fnal.gov/

LArSoft design principles and practices

- LArSoft is the primary production simulation, reconstruction for multiple experiments
  - A unique undertaking within HEP
  - Presents challenges of **design, coordination and organization**
- Central to the mission: a set of design principles, practices for the core software
  - Detector interoperability
  - Framework independence of data structure, algorithm code
  - Standardized interfaces, usage patterns
  - Modularization
  - Continuous integration
  - Tiered documentation
  - Peer code analysis and review
  - Centralized infrastructure support, coordination, policies and governance via the LArSoft Project

LArSoft design principles and practices

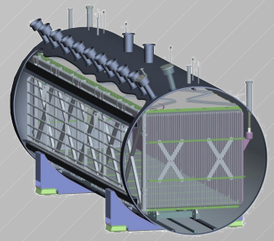
- LArSoft is the primary production simulation, reconstruction for multiple experiments
  - A unique undertaking within HEP
  - Presents challenges of **design, coordination and organization**
- Central to the mission: a set of design principles, practices for the core software
  - Detector interoperability
  - Framework independence of data structure, algorithm code
  - Standardized interfaces, usage patterns
  - Modularization
  - Continuous integration
  - Tiered documentation
  - Peer code analysis and review
  - Centralized infrastructure support, coordination, policies and governance via the LArSoft Project

Will touch on many of these in the balance of the talk

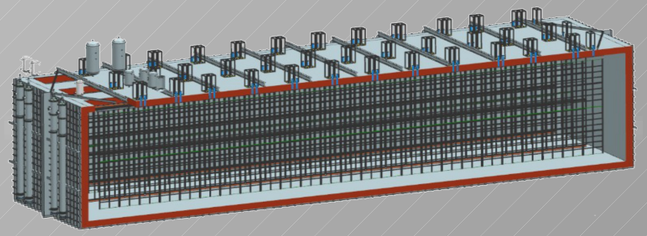
Detector interoperability

- The cornerstone of the entire project
- Rests on common features of LAr TPC geometry, physics, data

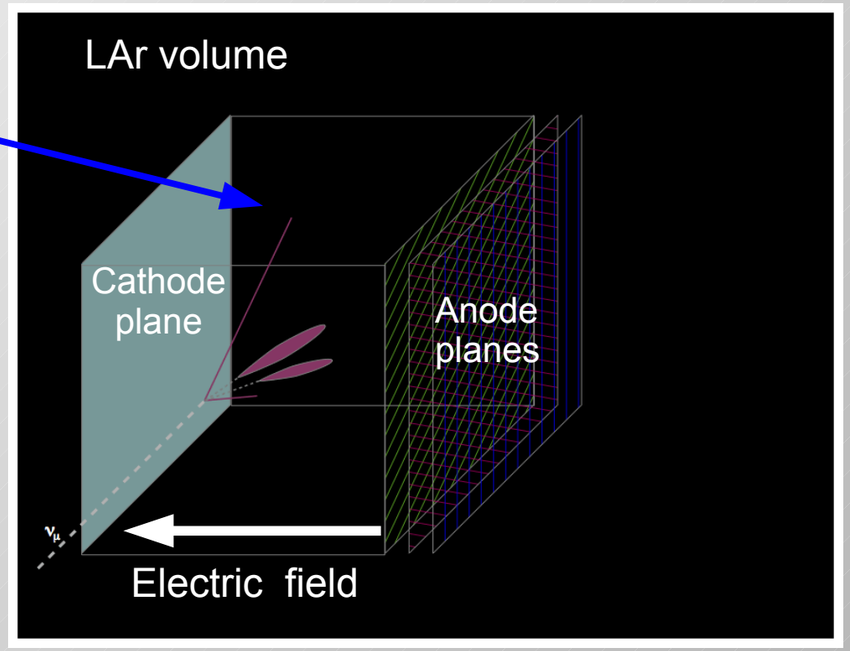
LArIAT
0.4m x 0.47m x 0.9m



MiroBooNE
2.2m x 2.5m x10m



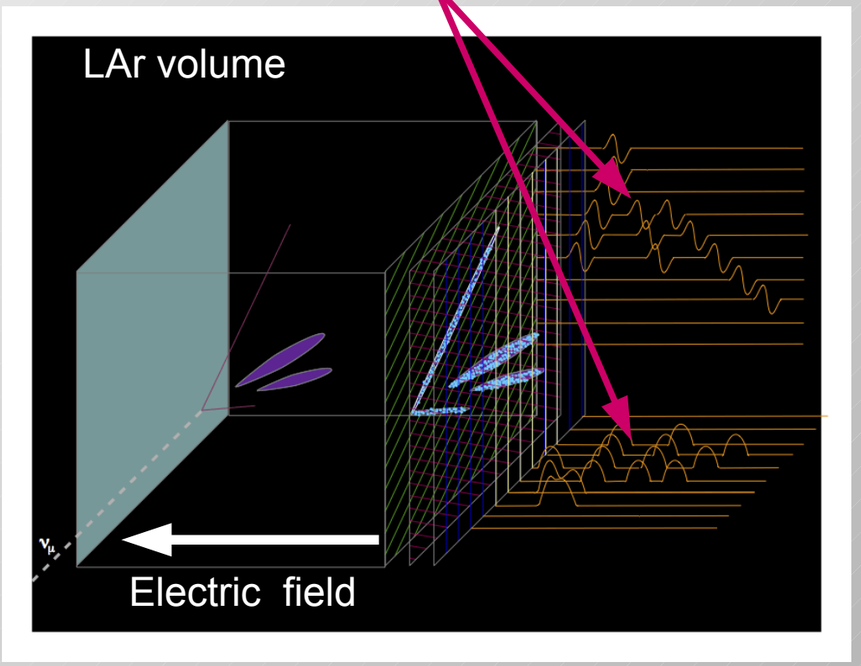
One DUNE far detector module
18m x 19m x 66m



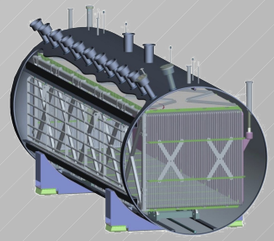
Detector interoperability

- The cornerstone of the entire project
- Rests on common features of LAr TPC geometry, physics, data

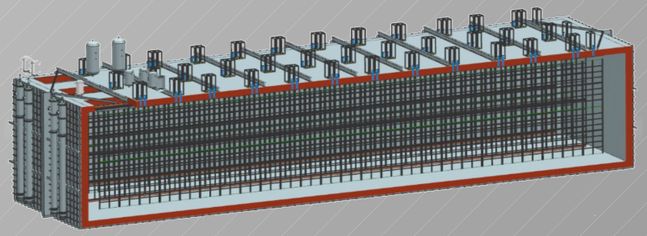
...Digitized waveforms in multiple views
induced by motion or collection of ionization...



LArIAT
0.4m x 0.47m x 0.9m



MiroBooNE
2.2m x 2.5m x10m

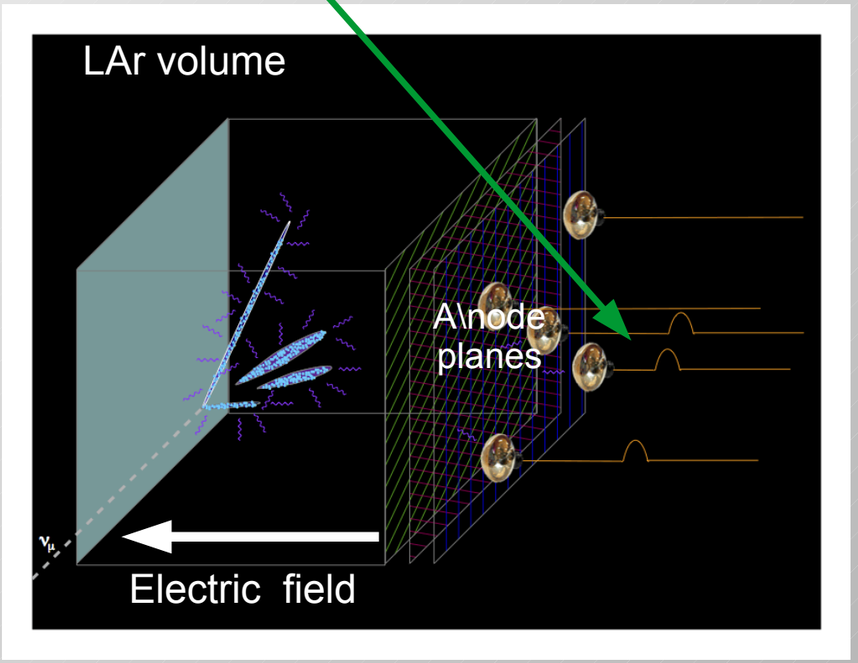


One DUNE far detector module
18m x 19m x 66m

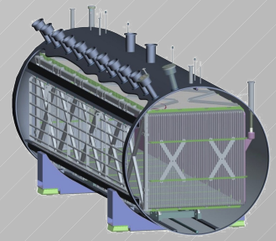
Detector interoperability

- The cornerstone of the entire project
- Rests on common features of LAr TPC geometry, physics, data

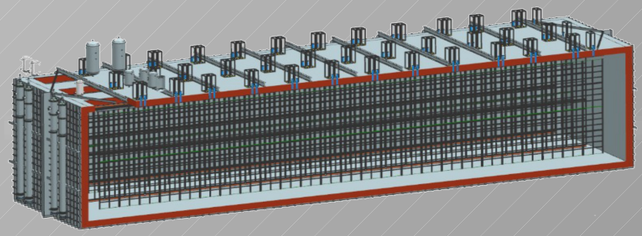
...Digitized waveforms from of detected
scintillation light from multiple photo-detectors...



LArIAT
0.4m x 0.47m x 0.9m



MiroBooNE
2.2m x 2.5m x10m



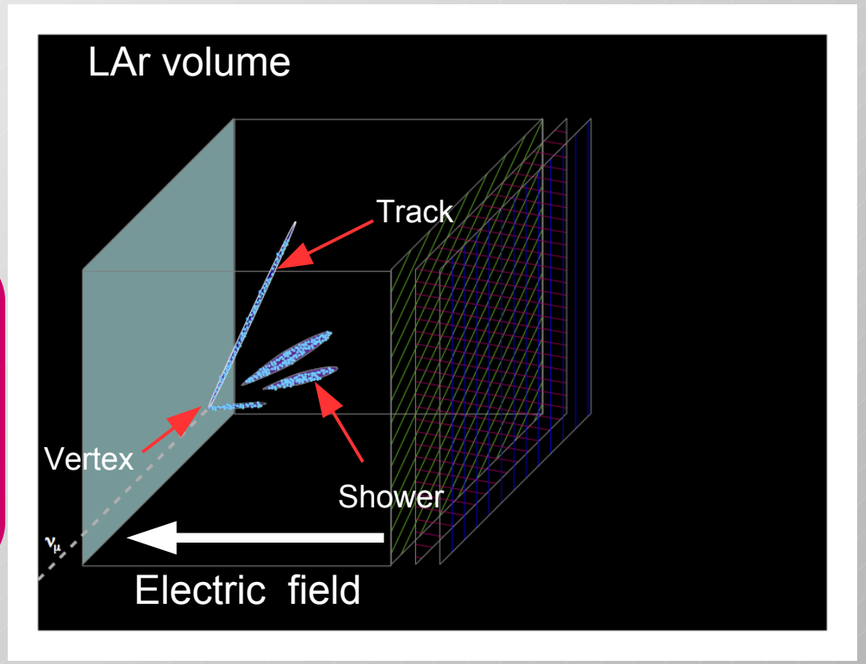
One DUNE far detector module
18m x 19m x 66m

Detector interoperability

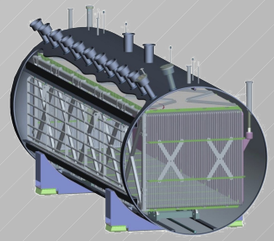
- The cornerstone of the entire project
- Rests on common features of LAr TPC geometry, physics, data

...That all reconstruction of 3D objects, measurements of their physical properties such as range and dE/dx

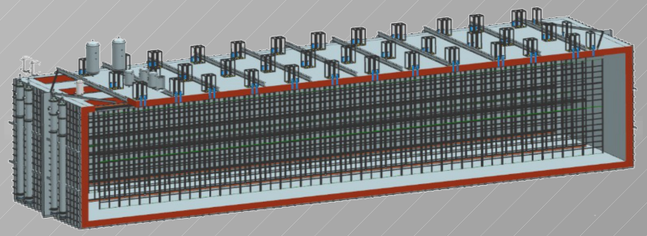
Allows definition of shared data structures, interfaces, workflow stages, and ultimately, shared algorithms, physics tools, utilities



LArIAT
0.4m x 0.47m x 0.9m



MiroBooNE
2.2m x 2.5m x10m



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

Detector interoperability

- Detector interoperability also requires
  - Use of 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
  - Shared interfaces to calibration, electric field maps, conditions information, etc.

Each must also allow detector-specific customization

Detector-specific elements

- Handle many detector-specific details via configuration
  - Geometry description
  - Generic detector properties
  - LAr conditions and properties
  - Photon transport / detection maps
  - Electric field map, etc.
- Detector-specific code currently required for
  - Raw data noise removal and signal processing
  - Electronics response in simulation and reconstruction
  - Simulation of raw data digitization

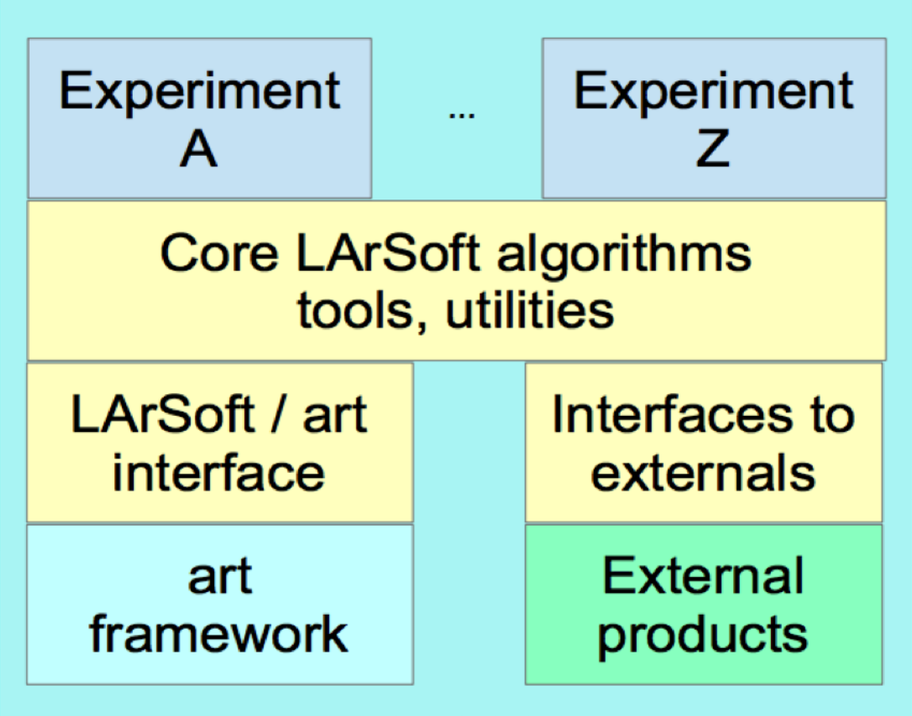
Detector-specific elements

- Handle many detector-specific details via configuration
  - Geometry description
  - Generic detector properties
  - LAr conditions and properties
  - Photon transport / detection maps
  - Electric field map, etc.
- Detector-specific code currently required for
  - Raw data noise removal and signal processing
  - Electronics response in simulation and reconstruction
  - Simulation of raw data digitization

Work in progress to create
a configurable framework
for sharing more of signal processing
and electronics response code

Framework independence

- Layered architecture



- Data structures, algorithms uncoupled from event-processing framework, external products
  - *art* modules serve as framework interface
  - Services designed to operate independently
- Event data model supported as external product

All of this allows

- **unit testing** throughout the suite for compliant code
- **broadening of toolkit** nature of LArSoft
- use of **alternative frameworks** (e.g., MicroBooNE's LArLite *)
- use of **minimal development environments** to facilitate easier / faster development cycles

* http://larsoft.org/larsoftlarlite-integration/

Standard interfaces, usage patterns

- Multiple algorithms available at each stage throughout the reconstruction workflow
  - Signal processing and hit finding within raw data waveforms
  - Clustering of TPC hits into 2D objects, and clustering of photo-detector hits into event times
  - Track and vertex finding
  - EM shower identification and energy estimation
  - Particle identification and momentum estimation
  - Event classification, etc...
- Communication between core LArSoft algorithms via centralized, common data objects
  - Use data structures to define shared interfaces for common workflow steps, tools, utilities when possible
  - Encourage layering algorithm workflows consisting of many smaller algorithms with same interface
  - Common patterns also make it easier to learn the code

Interfacing to external software products

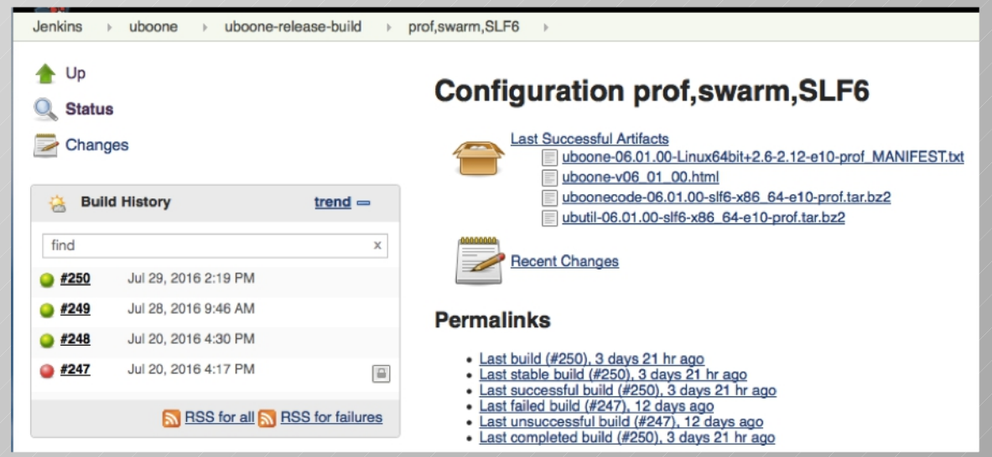
- Core LArSoft modules provide centralized common data structures, physics tools, shared algorithms
- Product APIs and common data products then used to construct direct interfaces to external software packages
- Some products currently integrated in this way:
  - Pandora pattern recognition software (see http://larsoft.org/pandora/)
  - Geant4 simulation (https://geant4.web.cern.ch/geant4/)
  - GENIE neutrino event generator (http://www.genie-mc.org/)

The development environment

- Open, centralized git repositories in Fermilab Redmine instance
  - Experiment-specific code lives in experiment-maintained repositories
- Source code build infrastructure based on system used for *art*
  - ups (Fermilab product versioning / environment configuration tool)
  - cmake
  - cetbuildtools / mrb (art build tools)
- Weekly integration releases to provide stable development platform
- Releases currently available for:
  - Scientific Linux 6 and 7
  - Mac OSX Mavericks and Yosemite
  - Ubuntu 14, and soon 16
- Distribution via a web site and cvmfs

Continuous integration and testing

- Operate centralized Jenkins continuous integration system that supports:
  - **Automated build and test programs** that run after **every commit** to the head of central code repository
  - **Automated email** to module owners with errors and warnings
  - Records of **memory and CPU usage** between versions
  - Supports **distributed and remote hardware** further testing



Coordination

- Review and coordination of contributed code via regular Coordination Meetings
  - Managed merging of code to head of repository
- In-depth code analyses for performance, practices, architecture with C++ experts
  - Targets identified in consultation with experiments, performed collaboratively with developers
- Regular gathering of requirements from the experiments, LArSoft community
- Regular meetings with experiment offline coordinators to discuss technical issues, short term core project work
- Regular meetings with experiments spokes to set overall priorities, goals, define direction for the project

Near / long-term plans

- Continue close collaboration between experiments, expanding the base of shared code
- Foster easier use/configuration of event displays, use of other visualization tools
  - Extend use of Paraview, Root, 2D, 3D and virtual environments
- Integration with additional external products, including:
  - BNL WireCell 3D reconstruction package (http://www.phy.bnl.gov/wire-cell/)
  - FLUKA detector simulation (http://www.fluka.org/fluka.php)
- Extend support for ProtoDUNE dual-phase detectors
- Add architectural extensions for machine learning algorithms
  - Such algorithms currently under active development in multiple experiments
- Review architecture in preparation for vectorization, multi-threading
- Continue improvements to usability of the code, interfaces, build system
- And more...

Summary

- LArSoft demonstrates a successful model of:
  - sharing primary LAr TPC simulation and reconstruction software across experiments
  - sharing of code with alternative frameworks, light-weight development environments

  with a common event framework, shared and experiment-specific algorithms
- The Collaboration includes ArgoNeuT, LArIAT, MicroBooNE, DUNE, SBND, Laboratory and University developers and scientists – and will welcome new experiments
- The collaborating experiments remain highly engaged in LArSoft at all levels
  - Providing new ideas and requirements
  - Contributing new code
  - Using and improving existing code shared by other experiments
- LArSoft is a vibrant community effort with many plans and ideas for future work

larsoft.org

- Introduction for new-comers at http://larsoft.org