The Liquid Argon Software Toolkit (LArSoft): Goals, Status and Plan

Ruth Pordes [*†]
Fermi National Accelerator Laboratory
E-mail: ruth@fnal.gov

Erica Snider
Fermi National Accelerator Laboratory
E-mail: erica@fnal.gov

LArSoft is a toolkit that provides a software infrastructure and algorithms for the simulation, reconstruction and analysis of events in Liquid Argon Time Projection Chambers (LArTPCs). It is used by the ArgoNeuT, LArIAT, MicroBooNE, DUNE (including 35ton prototype and ProtoDUNE) and SBND experiments. The LArSoft collaboration provides an environment for the development, use, and sharing of code across experiments. The ultimate goal is to develop fully automatic processes for reconstruction and analysis of LArTPC events. The toolkit is based on the *art* framework and has a well-defined architecture to interface to other packages, including to GEANT4 and GENIE simulation software and the Pandora software development kit for pattern recognition. It is designed to facilitate and support the evolution of algorithms including their transition to new computing platforms. The development of the toolkit is driven by the scientific stakeholders involved. The core infrastructure includes standard definitions of types and constants, means to input experiment geometries as well as meta and event- data in several formats, and relevant general utilities. Examples of algorithms experiments have contributed to date are: photon-propagation; particle identification; hit finding, track finding and fitting; electro-magnetic shower identification and reconstruction. We report on the status of the toolkit and plans for future work.

38th International Conference on High Energy Physics
3-10 August 2016
Chicago, USA

[*]Speaker.
[†]On behalf of the LArSoft Collaboration; FERMILAB-CONF-16-326-CD

1. Introduction

This section introduces the LArSoft [1] collaboration goals, scope and participants.

1.1 What is LArSoft

LArSoft [2] provides a common software architecture and toolkit for sharing physics data simulation, reconstruction and analysis for Liquid Argon Time Project Chamber (LArTPC) detectors and experiments. The LArSoft collaboration includes the experiments involved as well as the providers of the software infrastructure. The collaboration's mission is to develop fully automated processes for reconstruction of LArTPC event data, to share software and expertise in the algorithms and tools, and to provide a collaborative environment for the use of existing and new codes and methods. Representative physics outputs are shown in Fig. 1.

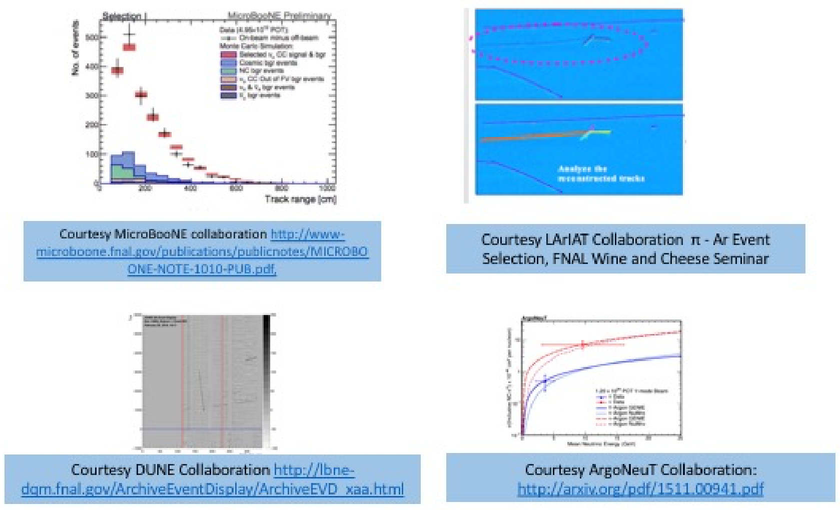

Figure 1: Examples of scientific output using LArSoft

1.2 Motivation for a shared software approach

Liquid argon detectors see very detailed information about each event—including many topologies typical of neutrino interactions that are difficult or impossible to resolve in traditional technologies (e.g. drift chambers). Each LArTPC provides essentially the same basic information after accounting for small detector differences. Algorithms developed for one experiment can therefore be used by another, as long as differences in geometry and detector response are properly accounted for. Aside from reducing the cost of software development for large experiments, such sharing enables the use of sophisticated infrastructure and algorithms by small experiments that otherwise lack the effort for large scale software development.

LArSoft currently includes at least one, and in many cases more than one, algorithm for signal processing, hit finding, identifying energy deposition clustering, shower finding, track finding, vertex finding, and particle identification among others. Furthermore, LArSoft algorithms benefit from the experience of multiple experiments.

1.3 LArSoft project

The LArSoft project coordinates the centralized software repositories, code acceptance and validation procedures, architecture, documentation of best practices and examples, as well as version release management and distribution. Configuration-time descriptions of the detector geometry, electronics, electric field, etc., drive the experiment-specific execution of their end-to-end workflows. The toolkit is coded in C++. It depends heavily on the *art* [3] event processing framework that is in use by many other experiments as well as those using LArSoft.

The *art* framework coordinates event processing via configurable, pluggable modules that add data to, and drop data from events, with conceptual components as shown in Fig. 2.

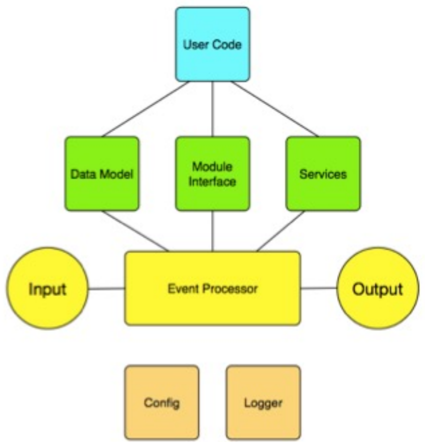

Figure 2: *art* components

The LArSoft software has a well-defined architecture based both on *art* and external components supplying capabilities such as particle generators, simulation, data displays etc. The layered architecture supports both common and experiment-specific algorithms and methods, as well as the integration with and interfacing to other packages. This architecture, outlined in Fig. 3, also facilitates the evolution of algorithms and environment to new methods (e.g. emerging Deep Learning techniques) and technologies (e.g. new high performance computing platforms).

1.4 Who Uses and Contributes to LArSoft

The ArgoNeuT [4], LArIAT [5], MicroBooNE [6], DUNE [7] and SBND [8] experiments currently use LArSoft, with the ICARUS [9] experiment considering some use of LArSoft as part of the Short Baseline Neutrino (SBN) program at Fermilab. The collaborations contribute algorithms and tools, as well as set requirements and priorities. The experiments provide validation of new releases of the software and define the physics goals and metrics. ArgoNeuT was the first experiment to use LArSoft, with members of the experiment being instrumental in moving existing methods from the ICARUS experiment into the LArSoft framework. MicroBooNE's first publications in the summer of 2016

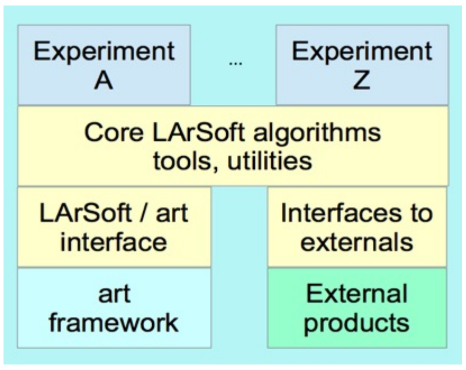

Figure 3: LArSoft layered architecture

included results from fully automated reconstruction of detector events. To meet this goal, the collaboration has driven much of the short-term development and support as the characteristics of their detector during data taking are fully explored. LArIAT, as a test-beam experiment, drives the approaches to correctly integrate and calibrate the TPC data with the many other ancillary detectors. DUNE has used and contributed to LArSoft for the DUNE 35ton prototype run at Fermilab in 2016. Work continues for DUNE and ProtoDUNE reconstruction and simulation, as well as ProtoDUNE data taking in 2018. SBND is coming up to speed and is benefiting from the algorithms already developed, before extending them to meet their specific detector needs. The

The Scientific Computing Division (SCD) at Fermilab provides the resources and services for the centralized code management and release activities. The core LArSoft project team also oversees the architecture and guiding principles of the whole system—encouraging and coordinating the means for sharing code, effectiveness of the contributions from the whole collaboration, and connections to external software packages.

External software packages currently depended on by and/or interfaced to the toolkit include those from the Pandora project [10], the *art* project, and the SCD simulation software groups. These groups also contribute to the overall architecture, data structure and interface definitions to enable a well modularized and extensible system.

A full set of requirements for the toolkit was developed at a workshop in 2015. As the experiments and software evolve, capabilities are mapped and matched to the recorded requirements. In addition, any new or modified requirements are captured. The current requirements document [11] includes more than forty authors who contributed to a broad range of topics, arising from the needs of their experiments.

2. Algorithms, Services and Data Products

This section describes the core components that enable multiple developers to contribute to a common set of software executables which run a variety of physics algorithms through configuration-driven workflows [12]. The executables that use the art framework, are physics programs where physics algorithms, provided as plug-in modules, are loaded and invoked in the order defined by run-time configuration files.

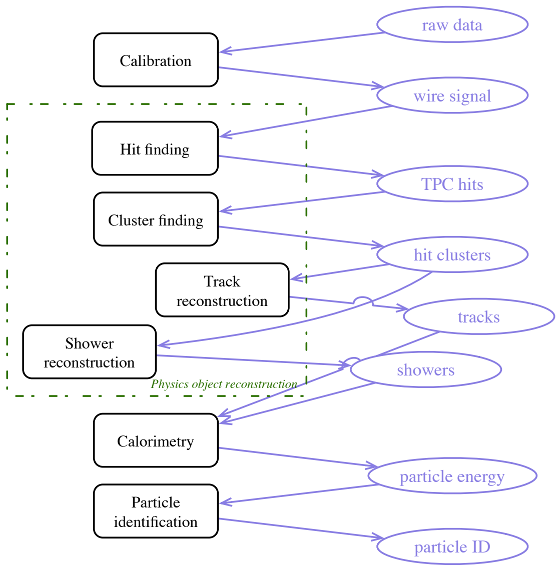

Figure 4: Sample reconstruction workflow

LArSoft provides abstract interfaces to retrieve the information needed by its algorithms, and at least a simple reference implementation, typically reading the information from program configuration files, databases, or local data files. Algorithm modules are also responsible for extracting the input data and storing the data output.

LArSoft code components can be grouped into broad functional categories: detector information; persistent data structures (data products); function execution operations; utilities and framework interfaces; graphical display; and examples [13]. LArSoft components are sequenced and combined into scientific data processing workflows defined using the *art* FHiCL [14] language where individual components interface to workflow steps and configurations used by any dependent products being used, such as Pandora, GEANT4 [15] and Genie [16]. The typical workflow is aligned to three main types of standard physics processes: simulation, reconstruction and analysis. Each user or experiments production group, defines the steps of each workflow or chain according to their needs, a typical example of which is shown in Fig. 4.

2.1 Algorithms

An algorithm in LArSoft (and *art*) is a C++ class that performs a single physics task or is part of a multi-class physics task. An algorithm communicates with the infrastructure through the LArSoft data products—both accessing these products and producing new ones. Typically methods of an algorithm class extract the parameters that drive its execution (from the configuration files identified with the run of the job), execute the physics method, have an output method, and are accompanied by one or more tests to check the correctness of the algorithm in a variety of workflow situations.

2.2 Services

A LArSoft service is a C++ class that performs a general operation in a single instance used by one or more LArSoft algorithms and/or one or more *art* modules. Examples of services are a random number generation and management service in *art*, and geometry and particle property files in LArSoft. Services are configurable using FHiCL which, for example, provides for interfaces to the detector specific properties and timings.

2.3 Data Products

The data products used and generated by such classes and generated by algorithms and services can be saved into *art* ROOT [17] files. LArSoft data products provide information that is transferred between algorithms, between algorithms and services. Translations between these data products and external software package protocols direct data exchange and module integration/interaction. For LArTPC data both the 2-dimensional and 3-dimensional information is crucial in order to properly understand the operation of the detector and the physics properties of the events. The data products describe both two-dimensional concepts, such as which energy depositions are closely correlated in both time and space, as well as complicated three-dimensional concepts such as which particle tracks and showers should be combined to describe a single particle interaction in the detector.

3. The Code

Both the experiment-specific and shared code resides in a set of git repositories together with scripts to build the different executables. Configuration FHiCL files provide for the workflow-specific steps and parameters to be used for the different code executions.

Included with the code are examples and documentation that provide patterns and guidelines for developing services and algorithms/modules. Guidelines include such aspects as encouraging developers to use common interfaces for detector and experiment-specific configuration information (e.g. detector geometry) and avoiding experiment-specific assumptions in the algorithms, e.g., the position of the first plane or wire, the wire spacing etc.

Table 1 shows the scale of the common/shared software repository as of August 2016. The experiment-specific code repositories are of equivalent sizes.

Table 1: Approx. Lines of Code

Type	Files	Cmnts	Code
C++	905	53K	190K
Header	758	41K	47K
CMake	164	597	4.5K
Perl	12	438	4K
XML	17	174	2K
Python	14	393	1K
SUM:	2K	96K	250K

3.1 Development Environment and Releases

The LArSoft build system ensures consistent builds among all supported platforms. The source codes in the git repositories are built, and releases made using ups Fermilab code versioning tool [18], cmake and cetbuildtool/mrb (the *art* build system); Documentation is automated through the use of Doxygen [19] and LXR [20]. The examples, as mentioned above, and a user-focused *art* workbook, support learning and communication of development patterns, available services, and best practices. The LArSoft team provides integrated, tested, supported weekly releases of the shared code, and help with experiment-specific code, that include changes in modules and services, transition to new versions of the underlying dependent external software packages—including operating systems etc.—and new capabilities. Multiple releases are supported to allow experiments to adopt them according to their internal schedules and priorities. In September 2016, LArSoft depended on ROOT 6, *art* V2.0, and Geant4 V10. The releases are distributed locally from a central website, and distributed to other institutions through the use of the CVMFS [21] software package. Releases are available for Scientific Linux (6, 7) [22], Ubuntu (14, potentially 16 in the future) and MacOSX (Mavericks, Yosemite).

3.2 Continuous Integration

A centralized service based on the Jenkins [23] framework supports automated build and test program execution after each code commit to the central repository—both for the shared code base and for the experiment-specific software. Errors encountered in the tests result in automated email being sent to the module owners. The memory and CPU usage of the tests, output parameter values and comparisons are made available through a web interface. The centralized service includes local test hardware but also supports test execution on remote sites to support OS and environments not available at Fermilab. This service has been in place for more than two years and regularly proves its value by finding issues before code is released into production.

3.3 Code Analyses

A focus of the past year has been to define and institute regular reviews of contributed code to improve the quality of the resulting code base and provide ongoing education to developers, many of whom do not have formal C++ or software engineering training. These analyses can be broad (architecture, design, implementation) or narrow (code readthrough) in scope and can take place at any time from when a new module or service is proposed, through to modules that have been in production for which performance or other issues are noted. The analysis process is structured in a series of steps as outlined in Fig. 5, with an emphasis on the commitment to the final step—follow up work—being discussed when the review is started [24]. We support use of several different performance measurement tools (igprof, valgrind, *art* memory and CPU time trackers) and provide expert consulting to interpret their output.

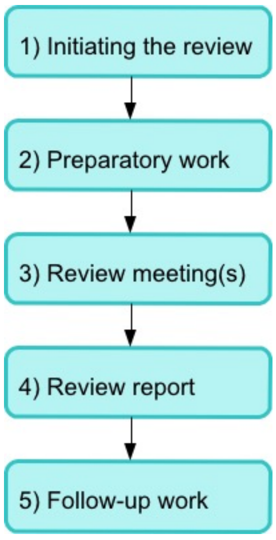

Figure 5: Analysis Process

The three module analyses conducted to date have resulted in constructive recommendations and have been well received by the module owners and the collaborations in general. While this work is resource intensive, there continues to be general agreement on its value.

4. The Future

As more data from more LArTPC detectors is collected, the event processing and analysis techniques need to become more sophisticated and accurate. Work will continue to provide good interfaces and integration with newly applied techniques such as machine/deep learning, image and signal processing. The increased use of High Performance Computing (HPC) systems by the High Energy Physics community and the evolution in computing hardware technologies available motivates revisiting code for effectiveness when running in multi-core and other environments. In collaboration with other experiments, the LArSoft community is looking to move away from the more parochial build tools towards the use of the more widely adopted SPACK [25] package for software build and distribution.

In other areas, work is continuing to improve the integration with the MicroBooNE lightweight analysis framework LArLite [26]. This is resulting in better support for other independent frameworks experiments may develop. Code analyses will continue on a regular basis with a goal to cover all new developments as well as the most used legacy code. Continued collaboration with the Pandora project is resulting in a more complete analysis and reconstruction system end-to-end—the aim is to reduce duplication of effort whereever possible. However, due to the many challenges of good anslysis of LArTPC events there is an increasing need to support multiple algorithms at any stage of the processing pipeline.

Additional work aims to foster development of more comprehensive and configurable event display and visualization tools—including extending the use of the Paraview visualization framework.

In conclusion, the LArSoft collaboration is energetic and the number of contributors to the code is steadily increasing. There is strong commitment from the experiments and software providers to common approaches and shared software as the needs and solutions evolve.

Acknowledgments

We thank the LArSoft Collaboration for its active contributions and support.

References

[1] http://larsoft.org

[2] E. D. Church, *LArSoft: A Software Package for Liquid Argon Time Projection Drift Chambers*,[arXiv:1311.6774]

[3] C. Green, J. Kowalkowski, M. Paterno, M. Fischler, L. Garren and Q. Lu,*The art framework*, J. Phys. Conf. Ser. **396**, 022020 (2012).

[4] R. Acciarri *et al.* [ArgoNeuT Collaboration], *First Measurement of Neutrino and Antineutrino Coherent Charged Pion Production on Argon*, Phys. Rev. Lett. **113**, no. 26, 261801 (2014) Corrigendum: [Phys. Rev. Lett. **114**, no. 3, 039901 (2015)] doi:10.1103/PhysRevLett.113.261801, 10.1103/PhysRevLett.114.039901 [
tt arXiv:1408.0598].

[5] I. Nutini [LArIAT Collaboration], *The LArIAT Experiment at Fermilab*, J. Phys. Conf. Ser. **689**, no. 1, 012020 (2016). doi:10.1088/1742-6596/689/1/012020

[6] K. Terao [MicroBooNE Collaboration], *MicroBooNE: Liquid Argon TPC at Fermilab*, JPS Conf. Proc. **8**, 023014 (2015).

[7] R. Acciarri *et al.* [DUNE Collaboration], *Long-Baseline Neutrino Facility (LBNF) and Deep Underground Neutrino Experiment (DUNE) : Volume 2: The Physics Program for DUNE at LBNF*, [arXiv:1512.06148].

[8] C. Adams *et al.* [LArTPC Collaboration], *LAr1-ND: Testing Neutrino Anomalies with Multiple LArTPC Detectors at Fermilab*, [arXiv:1309.7987].

[9] M. Antonello et al., *MicroBooNE and LAr1-ND and ICARUS-WA104 Collaborations*, arXiv:1503.01520 [physics.ins-det].

[10] J. S. Marshall and M. A. Thomson, *The Pandora Software Development Kit for Pattern Recognition*, Eur. Phys. J. C **75**, no. 9, 439 (2015) doi:10.1140/epjc/s10052-015-3659-3 [arXiv:1506.05348].

[11] https://cdcvs.fnal.gov/redmine/projects/lartpc-requirements/repository/revisions/master/entry/new-document/requirements.pdf

[12] Jim Kowalkowski, *Workflows and Workflow Systems at Fermilab* http://cd-docdb.fnal.gov/cgi-bin/RetrieveFile?docid=5551

[13] https://cdcvs.fnal.gov/redmine/projects/larsoft-architecture/repository/changes/output/LArSoftArchitecture.pdf?rev=master

[14] R. Putz *Specification of the Fermilab Hierarchical Configuration Language* https://cdcvs.fnal.gov/redmine/attachments/download/6639/grammar.pdf

[15] J. Allison *et al.*, *Geant4 developments and applications*, IEEE Trans. Nucl. Sci. **53**, 270 (2006).

[16] C. Andreopoulos *et al.*, *The GENIE Neutrino Monte Carlo Generator*, Nucl. Instrum. Meth. A **614**, 87 (2010) doi:10.1016/j.nima.2009.12.009 [arXiv:0905.2517].

[17] R. Brun, P. Canal and F. Rademakers, *Design, development and evolution of the ROOT system*,' PoS ACAT **2010**, 002 (2010).

[18] M. Votava, et al, *UPS UNIX product support* IEEE Seventh Conf Real Time 91 Computer Appl Nucl Part Plasma Phys. (pp. 156-159).

[19] http://www.stack.nl/ dimitri/doxygen/

[20] http://lxr.linux.no/+trees

[21] J. Blomer et al., *CernVM-FS: delivering scientific software to globally distributed computing resources*; Proceedings of the first international workshop on Network-aware data management. https://dl.acm.org/citation.cfm?id=2110217.2110225

[22] https://en.wikipedia.org/wiki/Scientific_Linux

[23] http://jenkins.io

[24] https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/Code_analysis_process_and_tools

[25] http://software.llnl.gov/spack

[26] K. Terao, *LArLite: C++ code development framework*, https://github.com/LArLight/larlite