LArSoft/LArLite Integration Report Status

Chris Jones, Marc Paterno

DRAFT which has not yet been reviewed by Kazu or the LArSoft Project

Activity to date

Interviewed Kazu Terao, Wes Ketchum, David Adams, Ryan Grosso, Corey Adams. Drafted report base on this.

Why do people use LArLite?

For the build mechanism, and development of algorithms

1. Faster build than when developing in LArSoft.
2. Smaller installation than needed for developing in LArSoft.
3. Faster installation with fewer steps than required for LArSoft.
4. Better stability; LArSoft's head is updated frequently, leading to breakage of user code.
5. Availability on Ubuntu.
6. Freedom to commit code to one's own repository, no need to synchronize with the LArSoft community.

For the event loop driver, doing analysis and developing (testing) algorithms

1. Event loop is simpler, thus faster, than the one in *art*.
2. Definition of data products can be experiment-specific.
3. Existing frequently-used data products in LArLite have a preferred interface to those in LArSoft, especially for iteration.
4. Modification of data products is easier; modified code can be committed to one's own repository without need for coordination between experiments or groups.
5. Integration with PyRoot for analysis or for writing tests is simple.

What is LArLite?

Build mechanism

The core of LArLite is a generator for producing a specific directory structure, GNU Makefiles and skeleton source files, including the necessary mechanisms to generate ROOT dictionaries. The design is such that there is a recommended way of combining the generated code with other code generated in the same manner. The structure and use of the Makefiles is not enforced in the system, which allows users to tailor it to their own personal needs.

Modular event loop driver

When many people speak of LArLite they are not referring to the build mechanism but instead to a modular event loop driver that is built using that mechanism. Algorithms are packaged into *modules*. The event loop driver reads events in sequence and passes the event data from module to module. The event loop driver does not have any mechanism for handling the concepts of Run or SubRun.

As part of the infrastructure for the event loop driver there are C++ classes which represent frequently-used data products. Some of these data products are designed to mirror LArSoft data products while others are experiment-specific.

Recommendations Span

- How to use LArSoft from LArLite
- How to develop, in LArLite, code that is interoperable with LArSoft
- Suggested changes to LArSoft and art
- Suggested changes to LArLite

Next Steps

- Discuss recommendations with Kazu and Erica.
- Discuss options with Panagiotis.
- Release full draft for comment and completion.