Proposed changes for art and supporting products

David Adams
BNL
January 15, 2015 v2

Introduction

DUNE is rapidly developing simulation, reconstruction and analysis software to assess its physics reach for a variety of detector options. The baseline design for the far detector system has four underground detectors each comprised of a cryostat filled with liquid argon (LAr) TPCs. There is a baseline design for the first cryostat to be installed in 2021 but there is still the possibility to change aspects of that design including wire spacing and angles and detector orientation. The remaining detectors will be delivered sequentially over many years and may have quite different layouts. In addition, DUNE will also construct prototype LAr TPCs and a near detector that may include liquid or gaseous argon TPCs.

To carry out studies as soon as possible, DUNE is making use of the LArSoft code developed at FNAL for other LAr TPC experiments. Having inherited code from its progenitor LBNE, DUNE is putting considerable effort into developing DUNE-specific code layered on top of LArSoft. There is urgent need to quickly develop this code to aid in the design of the early detectors. Somewhat in conflict, there is desire to ensure this same code can aid in the development of succeeding DUNE detectors and in the simulation of reconstruction of data from all detectors.

LArSoft makes use of art, an event-processing framework also developed at FNAL. Most of the DUNE studies are being carried out in the art framework but it is not certain that this framework will continue to be the ideal one for DUNE in the coming decades. In addition, there is the desire to develop interactive reconstruction environments to aid in the development of algorithms, e.g. where the current reconstruction is failing or slow. Ideally, the DUNE and relevant parts of the LArSoft code would be useable both inside and outside the art framework.

DUNE is a large collaboration and, rather than depend on a limited number of experts, the software environment should encourage a wide community to contribute and try out ideas. This implies the software should be modular so that a contributor and understand and contribute to one aspect without understanding the whole. The development environment should be user-friendly, i.e. natural and not require extensive training. Supporting infrastructure tools will naturally evolve in the coming decades (e.g. CVS to SVN to git) and it may be desirable to have interfaces that insulate users from such changes.

1. Make FCL comprehensible: Drop use of "@local::" in FCL. Drop prologs?

2. Make services including ServiceHandle accessible outside the art framework

3. Standardize services to have interface and single class for implementation

4. Add support for tools including ToolHandle

5. Add service to read and write event data

6. Provide module-like processing outside the art framework

7. Improve development environment:
   - simplify and speed up checkout/build (including package version conflicts)
   - simplify modification of package build instructions (including library lists)