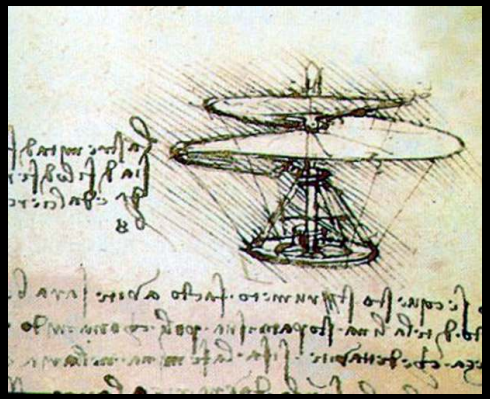

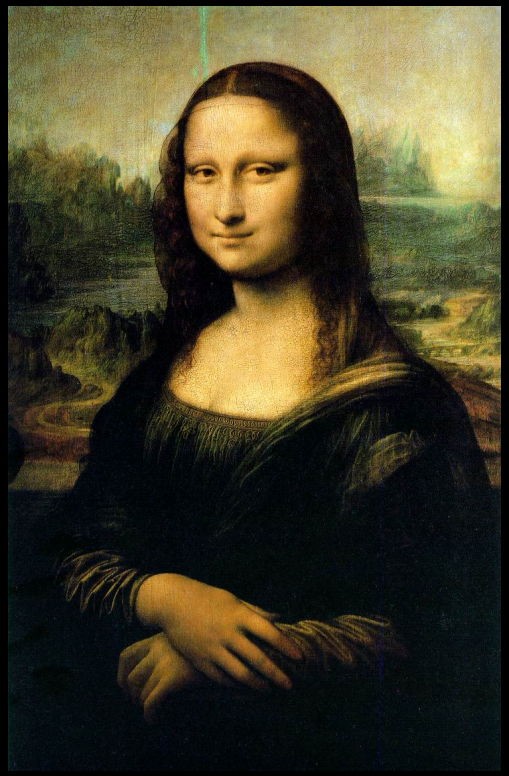

Overview of the art Framework

Chris Green, for the art team.
Common Infrastructure
Software Toolkit
Project Workshop.
24 Jan, 2014.

Fermilab
Fermi National Accelerator Laboratory
Office of Science / U.S. Department of Energy
Managed by Fermi Research Alliance, LLC

Users and User needs

User Base

- Intensity Frontier experiments:
  - **NOνA** (near detector beam data, far detector Q2 2014) and **NOνA**-DDT (with limited use of **artdaq**).
  - **Mu2e** (simulation).
  - **Muon g-2** (simulation).
  - **μBooNE** (data Q2 2014).
  - **LBNE** (planning, simulation).
  - **ArgoNeuT** (analysis).
- **Darkside/ DS50** (data with **artdaq**, analysis with **art**).
- Toolkits / packages used by multiple experiments:
  - **Nutools**.
  - **LArSoft**.
  - **artdaq**.
  - **IFDH**.
  - **artG4**.

Needs Met, Problems Solved

- Parallel multi-package build system – **MRB** (**Muon g-2**, **LArSoft**).
- OS support: SLF5 (and related), SLF6 (and related), OS X Mountain Lion.
- Framework as external binary product for experiments with little permanent programming expertise.
- Framework imposes as few constraints as possible on experiment's build system.
- Packaging and delivery of binary external products (compiler, **Geant4**, **ROOT**, **GENIE**, *etc.*) for development in a controlled environment. System privileges *not* required.

Needs Met, Problems Solved

- Framework for configurable modular simulation, reconstruction and analysis, with:
  - **ROOT**-readable output files.
  - Nested, experiment-defined hierarchy of scope for processing and data products (`Run`, `SubRun`, `Event`).
  - Runtime configuration and loading of modules and services.
  - Representation of related data "products" and collections of same for communication between modules.
  - Provenance and other metadata for data consistency and reproducibility checks.
  - Ability to run in grid environments.
  - Interface to **SAM** file lookup, retrieval and data mechanisms.
  - Ability to interface to (or form the basis of) experiments' event display.
  - Data integrity assured by being immutable once placed in the event.

art Components and Features

High-Level Components

Build and delivery.

- Delivery system is "**relocatable UPS**." **CVMFS** being developed, but not compulsory for experiments.
- **CMake**-based build system supports parallel builds. Used for all our packages, being adopted by some experiments. Distinct “install” step generates **relocatable UPS** products. Not necessary to use **art**: **Mu2e** uses **SCONS**, **LBNE** uses **worch**, other experiments or toolkit / package providers use **SoftRelTools**, **CMake**, or **MRB**.
- **MRB** allows simultaneous local builds of related packages against centrally-installed dependencies (*cf* “test releases”), while maintaining consistency of build (package ≡ repository ≡ **relocatable UPS** product ≡ smallest versioned unit).
- All **art**-suite packages are built closed-link (`--no-undefined`) and warning-free (`--Werror`) with no circular dpendencies.

High-Level Components

art suite.
Extensively reorganized after fork from **FWCore** *circa* March 2010 for dependency relationships and new features.

- **art**.
  Main framework.
- **messagefacility**.
  Event logging system, *long* history. Two different variants (standard, **NOνA**-online).
- **fhicl-cpp**.
  FHiCL is a configuration language designed to meet the requirements of intensity frontier experiments. Many discussions with stakeholders over the form and function of the language. Other bindings available.
- **cetlib** and **cpp0x** utility libraries.

Infrastructure and modularity

- Main application, including command-line option handling. Plugin use and workflow specified at runtime via **FHiCL** configuration read from file and post-processed according to command-line options.

```
#include "services.fcl"
process_name: DEVEL
source: { module_type: RootInput
  fileNames: [ "a.root", "b.root" ]
}
services: @local::services
services.user.ExptService: { par1: inf par2: 3 }
physics: {
  analyzers: { a1: {
      module_type: EMPerf
      useParticleID: false
      SelectEvents: { SelectEvents: [ "tp" ] } }
  }
```

Infrastructure and modularity

```
    filters: { ps1: {
        module_type: BlockingPrescaler
        blockSize: 3 stepSize: 200
        offset: 27 }
    }
    tp: [ "ps1" ]
    ep: [ "a1", "o1", "o2" ]
}
outputs: {
    o1: { module_type: RootOutput
          fileName: "raw.root" }
    o2: { module_type: RootOutput
          fileName: "pre.root"
          SelectEvents: { SelectEvents: [ "tp" ] } }
}
```

Infrastructure and modularity

```
# services.fcl
BEGIN_PROLOG
services: {
  user: { # User-defined services here.
        }
  Timing: { }
  SimpleMemoryChecker: { }
}
END_PROLOG
```

- Plugin generation and loading (product dictionaries, modules, services) via naming conventions (`libXXX_YYY_module.so`, `libXXX_ZZZ_service.so`, `libAA_BBB_dict.so`, *etc.*)—easier on build system.
- Modules: producers, filters, analyzers, sources, outputs.
- Intra-module parallelism via **Intel TBB**.

Infrastructure and modularity

- Producers and filters are restricted to trigger paths; analyzers and outputs are restricted to end paths. Disposition of defined paths is automatic.
- Services:
  - Open plugin / callback system used for I/O virtualization and geometry-related issues, also random number generation and other “non-physics” tasks.
  - Service interfaces and interface implementations; corresponding changes to service declaration and management system.
  - Services specified as `LEGACY`, `GLOBAL` or `LOCAL` in preparation for parallel operation.
  - Service callback mechanism simplified significantly and **SIGC++** dependency removed via use of C++2011 features (variadic templates). Local and global signals in perparation for parallel operation.
- Support for limited reconfiguration of modules and paths to support event display and DAQ applications.

Data Model and I/O

- Inputs / Outputs; **ROOT** file I/O; stream input, limited network streaming output of **ROOT** data.
- Class template for input sources.
- User-defined data products saved in `Event`, *etc.*. Support for non-persistable data products.
- References (`Ptr`, `View`) to items in collections. Limited support for polymorphism (`Ptr<Base>`). Compact persistent representation (`PtrVector`) of sequences of `Ptr` into the same collection. Conversion from `Ptr<T>` to `Ptr<U>`.
- Reorganized product-finding (`Group`, `EDProductGetter`, *etc.*) for greater efficiency, encapsulation and to support other requested features (`Assns`).

Data Model and I/O

- Bi-directional associations between items in collections (`Assns`, `FindOne`, `FindMany`, *etc.*).
- Product mixing (overlaid events). Data model complexities hidden from mix module authors by use of a module template and helpers.
- In-file **SQLite** DB for user and **art** metadata.
- `ParameterSet` (from **FHiCL**), stored in embedded **SQLite** database.

Other components and features

- Relying on and leveraging C++2011: `art::ValidHandle`, `std::shared_ptr`, `std::unique_ptr` and move semantics (including addition of products to the event), variadic templates for signals / slots, `static_assert` where appropriate.
- Histogramming service for modules (separate from event data).
- Ability to interface to event display, with random access to events.

Future work to meet user needs

Plans

- New OS support: OS X Mavericks, SLF7 (and related).
- New compiler support: Intel compiler suite, LLVM / Clang.
- Cross-compiling architecture support: ARM64, Intel Mic.
- Extend run and subrun processing features to meet experiment needs, including necessary changes to run / subrun concepts.
- Reduce "shape" restrictions on merging of data files where possible and appropriate.
- User-managed metadata and necessary improvements to metadata DB.
- Migrate metadata to **SQLite** DB to increase flexibility and simplify product access.

Plans

- Split processing of large events due to memory constraints.
- Upgrade `EventProcessor` and file modes to be more flexible amenable to (e.g.) DAQ use cases.
- Move toward pull rather than push output file open / close.
- `Ptr` into collections in subrun, and run-level products, and necessary improvements to metadata and product-finding.
- Work toward multi-schedule processing and thread safety.
- Streaming output.
- Migrate to C++2014.

Backup slides . . .

art Changes

How has art diverged from **CMSSW** since its fork (*circa* March 2010)?

- Simplification of inter-product references: removal of persistent type-erasing containers, `PtrVector` becomes more vector-like.
- `EventSetup` removed: implications to all plugin interfaces.
- Removal of one-file, two-file, system.
- **Python** configuration removed in favor of **FHiCL** configuration language. All parameters are tracked.
- `Schedule` rewritten, parts abstracted out.
- `EventProcessor` parts abstracted out (initial setup, *etc.*). Simplified to remove non-required operational modes.

art Changes

- End paths and trigger paths separated: analyzer / outputs constrained to end paths, filters / producers constrained to trigger paths.
- `InputSource` rewritten as pure interface.
- Source template allows easier writing of input modules by experiments.
- `FileIndex` overhauled.
- In-file **SQLite** DB. Parameter set info now stored here. Rest of metadata on the way.
- Build system: **CMake**-based, parallel-capable, including macros for easier specification of tests.

art Changes

- Plugin system replaced with one based on naming conventions.
- Reorganization of product-finding mechanisms (`Group`, `ProductGetter`, *etc.*): better encapsulation, realignment of concepts, interfaces and object management to clarify ownership and improve efficiency.
- Bi-directional associations `Assns`, `FindOne`, `FindMany`, *etc.*.
- Product-mixing interface.
- Flush principals.
- lumi $\rightarrow$ subrun, renamed types and functions throughout the system. `SubRun` now valid from 0 (lumi valid from 1).

art Changes

- Major library reorganization for dependencies (including **messagefacility**).
- **messagefacility** context setting moved to art.
- Closed-link builds; warning-free builds.
- Service interfaces and interface implementations; corresponding changes to service declaration and management system.
- Services are now `LEGACY`, `GLOBAL` or `LOCAL` in preparation for parallel operation.
- Service callback mechanism simplified significantly and **SIGC++** dependency removed via use of C++2011 features (variadic templates). Local and global signals in perparation for parallel operation.
- `ParameterSet` interface overhaul.

art Changes

- Limited support for reconfiguration of modules and paths under restricted circumstances.
- New application / command-line option handling system. Most command-line switches are converted to `ParameterSet` configuration during configuration postprocessing.