CS-Doc-5545-v3

Fermilab

Breaking the Silos: The *art* Documentation Suite

Rob Kutschke Fermilab Scientific Computing Division

CHEP 2015, April 16, 2015

Presented by: Chris Jones, Fermilab Scientific Computing Division

Introduction

- *art:* event processing framework used as an external product
  - In the same sense as ROOT, Geant4, CLHEP …
  - Mu2e, Muon g-2, NOvA, DarkSide50, MicroBoone, DUNE …
  - https://web.fnal.gov/project/ArtDoc/Pages/home.aspx
  - https://cdcvs.fnal.gov/redmine/projects/art/wiki
- A pressing need for integrated *art* documentation
  - Details for intermediates and experts
  - Onboarding materials for beginners
    - Self paced, self study (people start asynchronously)
    - *https://web.fnal.gov/project/ArtDoc/SitePages/documentation.aspx*
  - Reference manual
  - Useful for all experiments using *art*
  - Built around exercises that "just work"

Prerequisites and Co-requisites

- Prerequisites
  - Things we really can assume a user knows
  - Examples: elementary procedural programming, pointers.
- Co-requisites
  - Things that we need to discuss as they are encountered:
    - Some C++ features, Standard Library, ROOT, CLHEP, steps in building code, build system, git, unix environment, bash
    - What's an event? What's an event loop? What editor can I use?
  - Introduce it; give it a name so that people can look it up.
  - Describe what is needed for the task at hand.
  - Would prefer these to be prerequisites but it's not practical.
    - Product documentation often presumes significant prerequisites
    - Or it is not organized to suit our needs
    - It is siloed: each package is usually discussed in isolation.

Experience with Mu2e

- People who have experience on another experiment that uses modern HEP software learn Mu2e software rapidly
  - Mostly need to learn new syntax for well understood ideas
- People without this experience are often overwhelmed:
  - Very often the roadblock is in a prerequisite or a co-requisite
    - Many advanced features are used on day 1!
  - No existing way to learn the co-requisites in a reasonable amount of time.
  - One way to solve this is to integrate discussion of co-requisites into the onboarding materials.
- Very often senior people can mentor junior people in everything EXCEPT computing.
  - 20 years ago they could do that too

Main Elements of the Documentation Suite

- Introduction
  - Outline of the documentation suite; survey of prerequisites.
- Workbook
  - Onboarding for beginners; canned examples for others.
  - Co-requisites described as needed.
  - Self paced, self study exercises; must “just work”.
- Users Guide
  - Targeted at intermediates and experts; the “mother lode”.
- Technical Manual
  - Targeted at *art* maintainers and developers
- Reference Manual
  - LXR, Doxygen or similar
- Table of Contents, Index, Glossary

Everything cross-referenced

Status

- Introduction ~90% complete 120 PDF pages
- Workbook ~25% complete 260 PDF pages
  - Guess ~800 pages at completion
- User's Guide ~5% complete
  - Existing content is vacuumed up from experiments that use art.
  - Not vetted; not edited.
  - Designed as a reference, not as something you read from start to finish. Total size at completion O(1000) pages?
- LXR and git browsers available now.
- Other elements: not yet started

The *art* Workbook

- A sequence of exercises
  - Must "just work"
  - With explanatory text
    - Discuss co-requisites as they are encountered
  - Read; build; run; study the output; exercises plus solutions
  - Some exercises are to modify or extend behaviour
  - Some exercises are to understand and fix errors.
- Most of the early exercises are designed to be sequential.
  - Some later exercises are standalone.
- Exercises are built around a greatly simplified toy detector
  - Massless central tracker in a uniform solenoidal field
  - (We have a request to replace this with a simplified LAr TPC)
- Plan ~30 exercises; 8 available now.

The Biggest Lesson Learned

- First version of Exercise1:
  - Hits in the toy detector are represented by the class toy::Hit.
  - Get a collection of hits from the event
  - Print the event ID and the number of hits per event
  - Fill a histogram with the ADC value of each hit
  - No documentation of co-requisites at that time
- This crushed many people
  - It took days for many beginners to work through.
  - In almost all cases the stumbling blocks were:
    - Finding documentation for co-requisites
    - Missing cross-references to material previously discussed.
- In the end Exercise 1 was split into 8 exercises
  - Details in backup slides

Technology

- Code
  - Versioned; distributed as a readonly git repository
- External products
  - Versioned; available as a tarball for SL and OSX.
  - Installed on most Fermilab machines and on many machines at home institutions of *art* based experiments.
- Written material:
  - Versioned; matched to the code and external products.
  - LaTeX source managed by git; distributed as PDF.
  - Hyperlinked internal and external references
    - Modern PDF browsers highlight links and have a back button.
  - Will add other output formats if the tools are available.

Feedback From Users

- They like it a lot and want it finished.
- 2 to 4 days to skim the Introduction and work through the first 8 workbook exercises
  - Depends on which prerequisites and co-requisistes a user already knows and whether they try every exercise in detail.
- Many people are intimidated by ~400 pages
  - But it reads quickly: lots of source and output listings; instructions are repeated so that you do not need to flip around.
  - We need buy in from the senior people that a few days or even a few weeks is a valuable investment of their people's time.
  - We have buy in from some but others are looking for a unicorn.
- Guess: when the workbook is complete it will take 5 to 15 days to work through it in detail.

Timing and Staffing

- The plan is ambitious
  - Estimate ~2-3 FTE-years for the complete project
  - So far:
    - Domain expert effort is 100% volunteer
    - Integral of ~0.5 FTE-years
  - Calendar time to complete
    - ???? the volunteers have day jobs
  - Fermilab provided a part time technical writer
    - Outstanding in her role but not a domain expert
- Maintenance plan
  - Not yet developed
  - Expect that all exercises will be run as part of the test suite to certify a new version of *art*. Need to automate verification.

Meta-Questions

- Most people in HEP do NOT need to be computing experts.
  - What is the baseline skill set that most people should have?
  - What fraction of the community should be able to run jobs for their experiments?
  - What fraction of the community should be able write analysis modules for their experiment?
  - What fraction of the community should have the computing skills to contribute to algorithm development?
  - Is doing TTree analysis all that most people should need?
- HEP community has no answers to these questions.
  - The answers to these will inform the training materials we need to develop
  - Effort won't be assigned to training materials unless the community demands it.

Summary

- We have a plan for an integrated *art* documentation suite
  - Still a mostly volunteer effort; new volunteers welcome.
- Critical features
  - Usable by all experiments that use art
  - Integrated treatment of co-requisites
  - Cross-referenced; Table of Contents; Index; Glossary
- Would like, with permission, to link to or incorporate material describing prerequisites or co-requisites. Suggestions?
- For more information:
  - https://web.fnal.gov/project/ArtDoc/Pages/home.aspx
  - https://cdcvs.fnal.gov/redmine/projects/art/wiki
  - *https://web.fnal.gov/project/ArtDoc/SitePages/documentation.aspx*

Backup Slides

The class toy::Hit

```
namespace toy {

  // C'tor and other members elided for clarity

  class Hit {
  public:
    float adc() const { return _adc;}

  private:
    float _adc;   // ADC counts

  };

  typedef std::vector<Hit> HitCollection;

}
```

A Simple Module (1)

```
// Get the hits from the event and histogram the ADC value.
namespace toy{

  class ADCPlotter : public art::EDAnalyzer {

  public:
    explicit ADCPlotter(fhicl::ParameterSet const& pset );

    void beginJob() override;
    void analyze(art::Event const& event) override;

  private:
    art::InputTag _hitsTag;
    TH1D*         _hADC = nullptr;

  };
}
DEFINE_ART_MODULE(toy::ADCPlotter);
```

A Simple Module (2)

```
// Get the input tag from the run-time configuration
toy::ADCPlotter::ADCPlotter( fhicl::ParameterSet const& pset):
  _hitsTag( pset.get<std::string>( “inputTag” ))
{}


// Book the histogram
void toy::ADCPlotter::beginJob(){

 art::ServiceHandle<TFileService> tfs;
 _hADC = tfs->make<TH1F>( “hADC”, “ADC for Hits”, 32, 0, 32);

}
```

A Simple Module (3)

```
// Fill the histogram.
void  toy::ADCPlotter::analyze( const art::Event & event ){

 auto hits =
   event.getValidHandle<toy::HitCollection>( _inputTag );

 std::cout << “Event: “ << event.id()
    << “ has “ << hits->size() << “ hits.” << std::endl;

 for ( auto const& hit : *hits ){
   _hADC->Fill( hit.adc() );
 }

}
```

Ideas encountered in this example

- Basic framework ideas:
  – plugin-able modules, the run-time configuration system (parameter sets), services, data products, input tags
- art::Event, art::EventID
- The experiment specific classes:
  – toy::Hit, toy::HitCollection
- ROOT basics, including TH1D
- TFileService
- fhicl::ParameterSet, an image of the run-time configuration
- art::InputTag
- art::ValidHandle<T>
- DEFINE_ART_MODULE

Ideas glossed over on page 19:

- We have implicitly assumed that people already understand
  - What's an event? A run? A subrun?
  - What is the event loop?
- Intermediate level C++ language skills
  - Minimal knowledge of: classes, inheritance, templates
  - namespaces
  - std:vector<T>
  - typedef
  - Writing loops
  - Pointers, references, handles
- A module inherits from a base class.
  - You must override the analyze function
  - You may override some other member functions

Ideas glossed over on page 19:

- DEFINE_ART_MODULE is a directive C-Preprocessor
  - What's a C-Preprocessor
- auto is very, very confusing to beginners.
- What's a build system? What does it do? What's a link list?
- How to use git, which is used to distribute the example code.

Our Experience

- If you skip any of the material on the last two pages many beginners are completely confused.
- This example took days for beginners to get through

Exercise 1 turned into 8 Exercises!

- Exercise 1: Running an *art* job
- Exercise 2: Building and Running your First Module
  - Only the analyze member function
- Exercise 3: Some other Member Functions of a Module
  - Begin/End Run/SubRun/Job
- Exercise 4: A First Look at Parameter Sets
- Exercise 5: Making Multiple Instances of One Module
- Exercise 6: Accessing Data Products
- Exercise 7: Making a Histogram
- Exercise 8: Looping over Collections

Prerequisites and Co-requisites

- Prerequisites
  - Things we assume a user knows
  - Examples: elementary procedural programming, pointers.
- Co-requisites
  - Things that we need to discuss as they are encountered in our exposition of *art*
    - Introduce it; give it a name so that people can look it up.
    - Describe what is needed for the task at hand.
    - Do this once and reference to it from other places.
    - Often this drives us to split an exercise into 2 or 3 parts so that the earlier part(s) can discuss the co-requisites. The last part has the *art* content.
  - Many co-requisites we would prefer to assign as prerequisites but experience teaches we cannot.

The Plan for the Documentation Suite

Introduction
1. What is a framework
2. Define Prerequisites
3. Overview of documentation
4. …

Workbook

Exercise 1
Activity 1
Activity 2

Exercise 2
Activity 1
Activity 2

…

Exercise N

Reference Manual

Technical Reference

External Refs
Root, C++, STL, G4
…

Users Guide
Table of Contents
1. Users view of *art*
2. Modules
3. Services
4. FHiCL
5. Interface to G4
6. Interface to SAM
7. …..

Appendices
I. Best Practices
II. Trouble Shooting
III. CLHEP gap Filler Docs
IV. Glossary

Index

Status

90% Complete

Introduction

1. What is a framework
2. Define Prerequisites
3. Overview of documentation
4. …

25% Complete

Workbook

Exercise 1
Activity 1
Activity 2

Exercise 2
Activity 1
Activity 2

…

Exercise N

Reference Manual

Technical Reference

External Refs

Root, C++, STL, G4
…

5% Complete

Users Guide

Table of Contents

1. Users view of *art*
2. Modules
3. Services
4. FHiCL
5. Interface to G4
6. Interface to SAM
7. …..

Appendices

I. Best Practices
II. Trouble Shooting
III. CLHEP gap Filler Docs
IV. Glossary

Index