Breaking the Silos: The *art* Documentation Suite

Robert K. Kutschke
Fermi National Accelerator Laboratory, P.O. Box 500 Batavia, IL, 60542, USA
E-mail: `kutschke@fnal.gov`

Abstract. The *art* event-processing framework is used by almost all new experiments at Fermilab, and by several outside of Fermilab. All use *art* as an external product in the same sense that the compiler, ROOT, Geant4, CLHEP and boost are external products. The *art* team has embarked on a campaign to document *art* and develop training materials for new users. Many new users of *art* have little or no knowledge of C++, software engineering, build systems or the many external packages used by *art* or their experiments, such as ROOT, CLHEP, HEPPDT, and boost. To effectively teach *art* requires that the training materials include appropriate introductions to these topics as they are encountered. Experience has shown that simply referring readers to the existing native documentation does not work; too often a simple idea that they need to understand is described in a context that presumes prerequisites that are unimportant for a beginning user of *art*. There is the additional complication that the training materials must be presented in a way that does not presume knowledge of any of the experiments using *art*. Finally, new users of *art* arrive at random times throughout the year and the training materials must allow them to start to learn *art* at any time. This presentation will explain the strategies adopted by the *art* team to develop a documentation suite that complies with these boundary conditions. It will also show the present status of the documentation suite, including feedback the *art* team has received from pilot users.

1. Introduction

1.1. What is art*?*

art is an event-processing framework [1] designed for use by particle physics experiments and by other experiments with similar event-processing needs. Experiments use *art* to build programs that process data in a variety of contexts including software filters used as high-level triggers, online data monitoring, calibration, reconstruction, simulation, and analysis. In most previous particle physics experiments, infrastructure software, including the framework, was written in-house by each experiment, and each implementation has been tightly coupled to that experiment's code. This tight coupling has made it difficult to share infrastructure software among experiments, resulting in great duplication of effort.

art was created as a way to share a single framework across many experiments; each experiment uses *art* as an external product in the same sense that ROOT, Geant4 and CLHEP are external products. An experiment that uses *art* provides its experiment specific code to *art* via plug-ins. The *art* framework, and its anncillary tools, are developed and maintained by software engineers who are specialists in the field of particle physics infrastructure software. This provides a robust, professionally maintained foundation upon which physicists can develop the code for their experiments. Furthermore, the larger, more diverse user base of a common framework spurs innovation that benefits all of the experiments using *art*.

Since 2009, almost all new experiments at Fermilab have adopted art as a central component of their software suite. The list of experiments currently includes the Mu2e experiment, the Muon g-2 experiment at Fermilab, NOνA, ArgoNeut, MicroBoone, the DUNE 35T Prototype, LAr1-ND, LArIAT, and DarkSide 50. It is also being used for initial work on the DUNE experiment. An overview of Fermilab computing support for the Intensity Frontier experiments is available in these proceedings [2].

There are two other software toolkits that use *art* and that are designed for use by multiple experiments. LArSoft [3] is a toolkit containing algorithms for the simulation and reconstruction of events in Liquid Argon (LAr) Time Projection Chambers (TPC); it is being developed collaboratively by members of the many LAR TPC experiments. *artdaq* [4] [5] is a toolkit upon which an experiment can build a streaming data acquisition system. Many Fermilab Intensity Frontier experiments that are now in the planning or construction stage will use *artdaq*. These packages are mentioned for completeness and will not be discussed further in this paper.

1.2. The Audience for art *Documentation*
There is a broad spectrum of audiences for *art* documentation. At one end of the spectrum are the intermediate and expert users of *art*; often these physicists are intermediate or expert users of, or are developers of, their experiment's software. These physicists need concise, complete explanations of *art*'s features, organized so that they can quickly find answers to specific questions as they arise.

At the other end of the spectrum are physicists who are beginners in the software of particle physics: these physicists have little or no experience with *art*; they often have little knowledge of programming in general and of C++ in particular; many have not yet been introduced to the notion of an event, the event loop and other elementary concepts of particle physics computing. Moreover they may not yet have much familiarity with the experiment they are joining.

Between these extremes are experienced physicists who have not had the opportunity to keep their computing skills up to date. They are a valuable resource that it is important to better integrate into particle physics experiments.

The most difficult aspect of the documentation task is to address the needs of the physicists at the low end of the computing skill spectrum. This paper will present an overview of the plans for the *art* documentation suite and will then describe the work that has been done to develop materials targeted at beginners.

2. The Experience of Mu2e
Mu2e began to use *art* soon after the start of the experiment. At that time there was little *art* documentation available, just a few working examples, the *art* source code and friendly *art* developers. As the Mu2e software team learned *art*, they wrote documentation for themselves and that documentation remains available to new physicists joining Mu2e.

The experience of Mu2e is that newcomers to Mu2e who have both strong C++ skills and previous experience with modern particle physics software learn both *art* and the Mu2e software rapidly; for the most part they only need to learn a new syntax for familiar ideas. On the other hand, almost everyone without this experience was rapidly overwhelmed. There were relatively few people between these two extremes, although this may be an artifact of Mu2e being many years from first data. Other experiments that are closer to their first data have reported that they have more physicists between the extremes but that they too see a large number of people who have great difficulty learning *art* and their experiment's software.

The Mu2e software team worked with physicists who had difficulty learning the Mu2e software. After only a few such cases, a pattern was observed and this pattern has become clearer with time: the people who had difficulty simply did not have the necessary prerequisites.

Moreover the list of prerequisites is surprisingly long. Other experiments report precisely the same observation.

3. The *art* Documentation Suite

The *art* documentation suite is available online [6].

3.1. Prerequisites

In an ideal world the *art* documentation suite would begin by defining prerequisites and suggesting references that will allow users to acquire those prerequisites. The documentation could then be tightly focused on *art*. The obvious prerequisites are that new users of *art* should have C++ skills at the boundary between a beginner and an intermediate, including an understanding of the Standary Library; they should understand how to use ROOT and CLHEP; they should have some knowledge of unix, including scripting and the relationship between a shell and its subshells.

Experience with the Mu2e experiment showed that these obvious prerequisites imply a lower level of prerequisites: new users of *art* need to know how to operate in a terminal window based environment; they need to know how to use one of the standard unix editors; they need to know the difference between an interpreted and a compiled program; they need a basic knowledge of the steps needed to build a compiled program; they need to be able to visualize how a code fragment uses computer memory; they need to understand the general unix notion of an environment variable that describes a path. These are but a few examples of elementary computing skills that many new users of *art* have never learned. The absence of these skills must be acknowledged in the design of the on-boarding materials.

This list is long enough that it is simply not practical to give newcomers a set of references to read before they begin to study *art*; it would take far to long to work through them. Moreover, the Mu2e and *art* teams have been unable to locate appropriate, concise and stable references for many of the prerequisites. One idea was to recommend selected chapters or sections from various books and online resources. However this needs to be done very carefully: if the instructions tell users to read chapters 1, 5, and 15 of some reference, it is often not possible to understand the material in chapter 15 without having first read many of the prior chapters. So the short cut of giving selected readings is often illusory. In additional the *art* team is reluctant to make references to web sites unless there is a reasonable expectation that the links to that web site will still work in a few years. For example, http://www.cplusplus.com has proved both stable and valuable.

Another weakness of simply defining prerequisites is that the existing documentation for prerequisites is siloed, that is the ROOT documentation describes ROOT in isolation, and son on. On the other hand, particle physics code frequently makes use of many prerequisites on a single line. When a user is presented with this their first task is to parse that line to determine which parts of the line refer to *art* ideas and which parts of the line refer to ideas from the prerequisites. This is automatic for experienced users but it is one of the great difficulties for new users. One of the key functions of documentation for beginners is to teach them how to recognize which elements of the code come from which of the prerequisites. This is the sense in which the documentation must be integrated.

In short, there is a wealth of documentation about the prerequisites but it is not organized to suit the needs of a newcomer to an experiment that uses *art*. The *art* documentation must supply this organization.

3.2. Co-requisites

The previous section argued that carefully defining prerequisites will not, by itself, solve the problem of on-boarding beginners in particle physics computing to *art* and their experiments's software. The solution is to recast many of the prerequisites as co-requisites. This is in analogy to a quantum mechanics professor and a mathematical physics professor cooperating to ensure that users are taught Associated Laguerre polynomials just before they learn about the hydrogen atom.

As an idea from the co-requisites is needed in the exposition of *art*, the *art* documentation describes it briefly, with a narrow focus on the task at hand, and gives it its proper name. The *art* documentation then refers the reader to the co-requisite's own documentation for any additional information. When appropriate, the *art* documentation advises the reader than the topic at hand is quite large but that a complete understanding is not necessary at this time.

There remain some true prerequisites and the Introduction section of the *art* documentation suite discusses them at some length. For example, familiarity with procedural programming C++ is a prerequisite and the Introduction reviews those features that the reader must understand. On the other hand an understanding of inheritance and templates is not a prerequisites; these are discussed as co-requisites because both are enormous topics from which a beginner need only understand a few paragraphs. In the Introduction the user is advised that if they know the reviewed material well they will be ready to learn *art*; if they are not familiar with a few of the ideas they should proceed but they will need to do some additional homework; if they are not familiar with a lot of the material, then they should learn it before continuing to learn *art*.

The key to making this strategy succeed is to work with the incoming users to understand what can be assigned as a prerequisite and what must be treated as a co-requisite.

Clearly the effort required to develop discussions of the co-requisites is large. It would not make sense to do this for a most individual Intensity Frontier experiments viewed in isolation — it would be less effort to mentor individuals who can mentor others in turn. But *art* is used by a large, and growing, community of experiments. The authors believe that size of the community justifies the effort: if newcomers to the experiment first learn *art*, along with the co-requisites, they will be ready to learn their experiment's software quickly.

3.3. Workbooks vs Schools
It takes a large effort to prepare and run a particle physics computing school. Therefore such schools are run infrequently. On the other hand, new users join experiments continuously throughout the year; even during the summer peak, the starting dates for new students extend from May 1 to August 1. So there is no ideal time to run a school.

The preferred solution is that the material to on-board beginners be organized as a self-paced, self-study workbook. In this way new users can being their on-boarding as soon as they are ready, whether that be at the lab or at their home institution.

3.4. Overview of the art *Documentation Suite*
When complete, the *art* documentation suite will have six main components:

- Introduction: an outline of the documentation and a review of the prerequisites.
- Workbook: self-paced, self-study exercises for beginners; discussion of co-requisites.
- Users Guide: the "mother lode", targeted at intermediates and experts.
- Reference Manual: LXR, Doxygen or similar.
- Technical Manual: internal details targeted at developers and maintainers of *art*.
- Table of Contents, Glossary and Index:

All of the above must be cross-referenced. For example, someone working through a Workbook exercise on handles to data products will only be told what is necessary to use a handle in typical day-to-day usage; they will be referred to the appropriate section of the Users

Guide for full details. On the other hand, that section of the Users Guide can refer to the Workbook for a working example. This minimizes multiple points of maintenance. But it does require coordination between the design of the Workbook and the design Users Guide.

At the present time the introduction is about 90% complete and fills about 120 pages of a PDF file. The Workbook is about 25% complete and fills about 260 pages of a PDF file; the authors estimate that it will fill about 800 pages at completion. The Users Guide is about 5% complete; the existing content was vacuumed up from assorted experiments that use *art*; it is not yet edited or vetted. The Users Guide is designed as a reference work, not something that one would read from beginning to end; the authors estimate that it will fill about 1000 PDF pages at completion. LXR and git browsers are already available for the *art* code and for the code of many of it ancillary tools. The Technical Manual has not yet been started. The Table of contents is present; the existing material has been indexed and scanned for appropriate glossary entries.

3.5. The Workbook

The primary purpose of the Workbook is to onboard new users, from beginners in particle physics computing to experts coming from other experiments.

The Workbook is organized as a sequence of self-study, self-paced exercises accompanied by explanatory text; each exercise must “just work”. Most of the early exercises are designed to be done sequentially, while many of the later exercises are standalone and may be done in any order. About 30 exercises are planned and 8 are available now.

The exercises are built around a greatly simplified toy detector, a massless cylindrical tracker in a uniform solenoid field. One of the reasons behind this choice is that it did not show favoritism to any of the experiments now using *art*, none of which have a cylindrical tracker. There is also a very good reason not to use actual code from any real experiment in these exercises: such code has features needed to address the complexities of a real experiment; frequently those complexities will obscure the point of the exercise and will confuse anyone who is not already an expert in that experiment. Moreover such code will change with time, necessitating a rewrite, possibly major, of the accompanying text. The decision to build the exercises around a toy detector is firm.

The *art* team has received a request to replace the toy central detector with a toy LAr TPC; this request is under discussion with the stakeholders.

Users of the Workbook will checkout code from a git repository; for each exercise, they will build code, run code and inspect its output. The accompanying text will discuss the features of *art* that are illustrated by the code along with any co-requisites that are needed to understand the code. Each exercise concludes with some optional tasks for the user to try. Some tasks are to extend or modify behaviour seen earlier in the exercise; in each case a solution is provided and discussed. Other tasks are to debug a piece of code that fails to compile, gives a run-time error or produces incorrect output; again solutions are provided.

Because of the material provided for beginners, the Workbook is long. New *art* users who are experienced C++ programmers and are familiar with modern particle physics software should just skim the Wookbook and choose where to read more deeply and where to work through the exercises. The main goal for such users is to acquire the layout of the documentation and to learn the names that *art* uses so that you can look up details when necessary. True beginners should read everything and work through every exercise. Those in between should skim the Workbook but presumably will find it necessary to do more of the exercises than will experts.

3.6. An Important Lesson Learned

An important lesson learned was to make each exercise small enough that most beginners could get through it in a few hours. The first version of Exercise 1 was conceived as follows:

- Hits in the toy detector are represented by the class toy::Hit.
- Get a handle to a collection of hits from the event.
- Print the event ID and the number of hits per event.
- Limit the printout using a run-time configurable parameter.
- Fill a histogram with the ADC value of each hit.
- Change the number of events processed.
- Change the input file.
- Read both input files in one job.
- Write an output file and read it back.

When this was first deployed, the documentation did not include a discussion of co-requisites. Intermediates and experts were able to work through this quickly. But this exercise crushed many beginners; after several days they had not completed the exercise. There were two sorts of stumbling blocks. First, users recognized that they were missing a prerequisite and they spent a long time searching, inefficiently, for information about that prerequisite; often the hardest part was to determine which prerequisite to read about. Second, at that time the documentation was not cross-referenced; so users frequently had to search for material they had seen before but not yet memorized.

In the current version of the Workbook, with a full discussion of co-requisites, this single exercise was split into 8 exercises, each with a narrow focus.

4. Technology

The code used by the Workbook is versioned and is maintained in a git repository. Users have readonly access to the repository and the Workbook does not teach users how to contribute to a git project. This is a carefully considered decision because not all experiments use git and those that do use git have adopt different git workflows.

To run the exercises, one needs access to a build of *art* and its tool chain. This too is versioned. Complete sets of binaries, including the compiler, are available as web-visible tarballs for many versions of Scientific Linux and Mac OSX. For those working on experiments using *art*, the binaries are already installed on your experiment's machines.

The written material is also versioned and each release of that material is matched to a particular version of the Workbook code and a particular version of *art*. The written material is maintained as LaTeX source files stored in a git repository and the material is distributed as a single PDF file. Distribution as a single PDF file allows full searching. The PDF files use hyperlinks to provide internal cross-referencing and to refer to outside material. Most modern PDF browsers have a back button, which amplifies the power of the cross-referencing.

The *art* team is interested in evaluating other output formats for the written material. One of the boundary conditions is that the content must be strongly versioned and it must be possible to develop a bug-fix branch off of an old version. In addition the final product must be searchable and the cross-references must be hyperlinked. Finally the candidate technology must support the automatic updating of cross-references when new chapters and sections are inserted.

5. Feedback from Users

The *art* team has asked for feedback from everyone who has used the Workbook and has received detailed feedback from about 6 pilot users with broad range of skill in the prerequisites; the pilot users do not include absolute beginners; they do include a student who is new to particle physics but who has strong computing skills. The short answer is that all of these users like the Workbook a lot and they would like to see it finished. A former CMS collaborator commented. "I finally understand what I am doing when I use the CMS software."

Most people took 2 to 4 days (possibly over the course of several weeks) to skim the Introduction and work through the first 8 exercises. Those with a higher skill level in the prerequisites completed the work faster.

Many people report that people are intimidated by the sheer size of the PDF file; the Introduction and Workbook are currently about 400 pages. But once you start you will discover that it reads quickly because there are many source and output listings; in addition, detailed instructions are often repeated so that users do not need to flip through the previous chapters to remember how to run an exercise.

The Mu2e collaboration has another plan to mitigate the intimidation factor. The plan is to assign some number of exercises of the Workbook and to follow that up with exercises using Mu2e code. This will be followed by some more Workbook exercises and some more Mu2e exercises. A person following this sequence does not need to follow it until the end; they can put it aside as soon as they have learned enough to start work on their initial project.

The current best guess is that, when the Workbook is complete, it will take 5 to 15 days to work through it in detail. It is important to convince the particle physics community that an investment in this scale is both necessary and appropriate. Some senior members of the particle physics community agree with this position but others still say that the want their summer student, who has never programmed before, to be an effective *art* user in less than one week. This simply won't happen; it is possible to find ntuple/TTree based projects for such students but it is not possible to teach them both *art* and their experiments software in less than one week.

6. Timing and Staffing

The complete *art* documentation suite is an ambitious project. The current estimate is that it will take 2 to 3 FTE-years of domain expert time to execute the complete project. To date the domain expert effort has been 100% volunteer and integrates to about 0.5 FTE-years. The calendar time to complete is unknown; the volunteers have day jobs. Fermilab has provided a part time technical writer who is outstanding but who is not a domain expert so she cannot create content or certify contributions for correctness.

The authors of the documentation suite recognize that a maintenance plan is required. A plan has not yet been developed but one will be as the project moves forward. One of the first steps will be to automate the testing that the exercises continue to work as designed; these tests will be incorporated into the pre-release testing for new versions of *art*.

7. Some Meta-Questions

The particle physics community understands that not every particle physicist needs to be a computing expert. However the community does not have a coherent understanding of what distribution of computing skills is necessary; this includes questions such as:

- Is doing a TTree analysis the only skill that most particle physicists should need?
- If not, what is the baseline computing skill set that most particle physicists 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, including commissioning, calibration, reconstruction and simulation?
- What fraction of the community should acquire the skills needed to develop algorithms that exploit the new features of highly parallel architectures?

Understanding the answers to these questions will inform the training materials that the community needs to develop.

8. Summary

This paper has presented a plan for an integrated *art* documentation suite that will be usable by all experiments that use *art*; it will contain an integrated treatment of co-requisites; it will be build around coherent set of code examples that can be discussed both by the Workbook and by the Users Guide; it will be cross-referenced and have navigation aides including a Table of Contents, an index and a glossary.

The Introduction section is almost complete and the Workbook section is about 25% complete. This is already enough material that the Workbook can form the basis for on-boarding new users. Users who have mastered the exercises of the *art* Workbook will be well prepared to learn the experiment's *art* based software.

This has been mostly a volunteer effort and new volunteers are welcome. The time to completion is uncertain because the volunteers have day jobs.

If you are aware of good material the describes the prerequisites or the co-requisites, please let us know. The *art* team would like, with permission, to link to this material.

References

[1] https://web.fnal.gov/project/ArtDoc/Pages/home.aspx
[2] Group, Craig, "Fermilab Computing at the Intensity Frontier", in these proceedings.
[3] https://cdcvs.fnal.gov/redmine/projects/larsoft
[4] https://cdcvs.fnal.gov/redmine/projects/artdaq
[5] Biery, Kurt, *et. al.*; "Recent Developments in the Infrastructure and Use of *artdaq* ", in these proceedings.
[6] https://web.fnal.gov/project/ArtDoc/SitePages/documentation.aspx