

Balancing agility and gate-keeping

The LArSoft development model

Erica Snider

Fermilab

June 22, 2016

The tension in code development

- Two competing desires common in code development
 - Ability to introduce new ideas, new features quickly
 - Minimal time spent in design
 - Low barriers to writing code, getting into releases
 - Just make it work!
 - Writing production quality code
 - Well tested
 - Well thought-out, integrated design
 - Usually considerable re-writing en route



Like many software projects, LArSoft benefits from both attributes in ample quantities

The tension in code development

- Two competing desires common in code development

- Ability to introduce new ideas.

- new features

- Low b
 - into r

- Minim

- Just n

- Writing p

- Well t

- Well t

- Usual
 - enrou

Both important to 'usability'

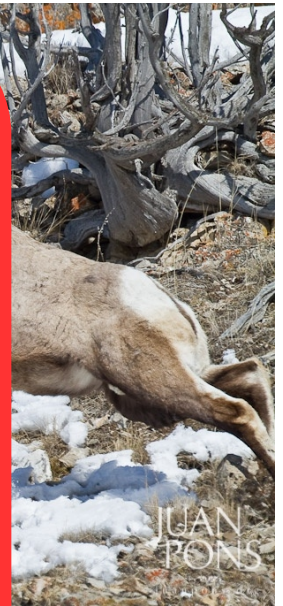
Users depend on 'production quality'

Developers on 'agility'

*But also 'production quality' as a
foundation for development*

So need to examine the LArSoft
development model

Like many software projects, LArSoft benefits from both attributes
in ample quantities



The LArSoft development model

- Documented on the LArSoft wiki:
https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/Developing_With_LArSoft
- The steps
 - Designing
 - Decide how to structure the solution
 - Writing code
 - Standards!!
 - Building
 - mrb, cmake, maybe cetbuildtools and its configuration for art, LArSoft
 - Testing
 - Documenting
 - Getting code into a LArSoft release
 - Must follow a procedure

The LArSoft development model

For our purposes today, only care about these

- Documented on the LArSoft wiki:
https://cdcv.sfnal.gov/redmine/projects/larsoft/wiki/Developing_With_LArSoft
- The steps
 - Designing
 - Decide how to structure the solution
 - **Writing code**
 - **Standards!!**
 - Building
 - mrb, cmake, maybe cetbuildtools and its configuration for art, LArSoft
 - Testing
 - Documenting
 - **Getting code into a LArSoft release**
 - Must follow a procedure

Contributing code

- How do we go about contributing code to LArSoft?
 - Create a branch in some repository
 - Create / modify and test code
 - Next steps depend on the type of change
 - Changes that do not affect behavior
 - Just merge into develop
 - New code or features that do not change dependencies, bug fixes
 - Merge into develop
 - Discuss at LArSoft Coordination Meeting to make people aware of the change
 - New code that introduces new dependencies, that breaks existing code or data, that alters behavior
 - Discuss at LArSoft Coordination Meeting
 - Upon approval, LArSoft team merges into develop during release creation procedure
 - Weekly integration releases to incorporate changes

Contributing code

- How do we go about contributing code to LArSoft?

- Create a branch in some repository
- Create / modify and test code
- Next steps depend on the type
 - Changes that do not affect behavior
 - Just merge into develop
 - New code or features that do not change dependencies, bug fixes
 - Merge into develop
 - Discuss at LArSoft Coordination Meeting to make people aware of the change
 - New code that introduces new dependencies, that breaks existing code or data, that alters behavior
 - Discuss at LArSoft Coordination Meeting
 - Upon approval, LArSoft team merges into develop during release creation procedure
- Weekly integration releases to incorporate changes

In most cases, a discussion of some sort at the LArSoft CM is required

Prior to integrating code, additional work may be requested.

What you need to know first

- Policies, guidelines and standards at all levels
 - LArSoft design principles

LArSoft concepts (on larsoft.org)

Design principles (on larsoft.org)

Architecture document

What you need to know first

- Policies, guidelines and standards at all levels
 - LArSoft design principles
 - Coding guidelines

LArSoft

- Coding guidelines and conventions
- Guidelines on writing/using services
- Guidelines on writing/using algorithms

art

- art module design guide
- Data product design guide
- Guidelines for the use of pointers

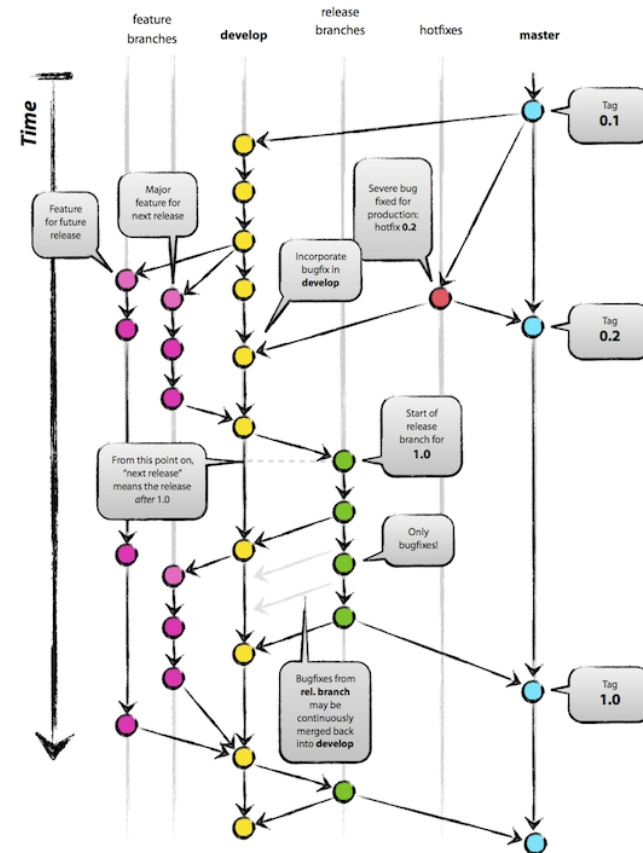
C++

- Lots of online resources...

What you need to know first

- Policies, guidelines and standards at all levels
 - LArSoft design principles
 - Coding guidelines
 - **Git branching model**

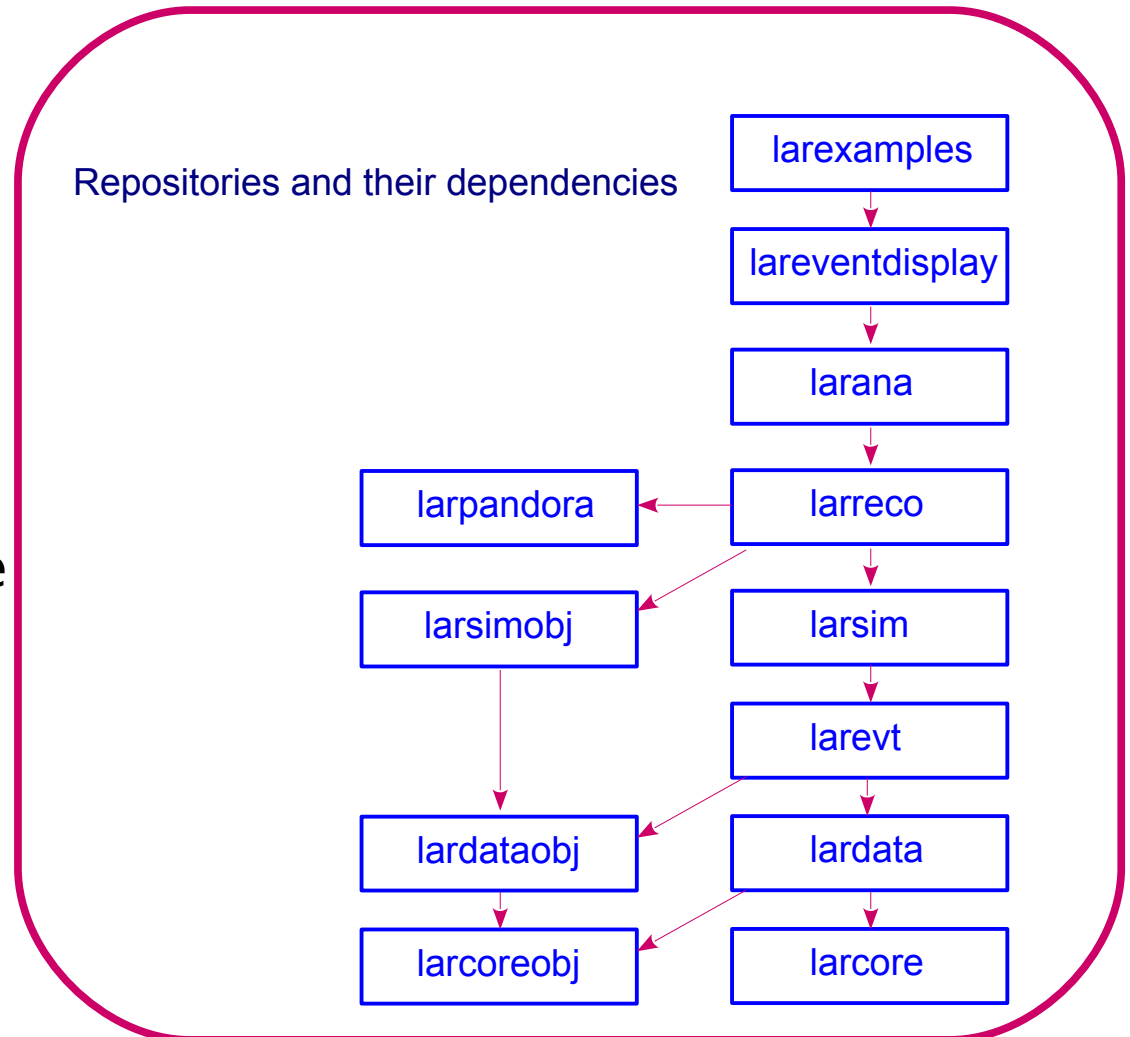
LArSoft git guidelines / branching model



Author: Vincent Driessen
Original blog post: <http://nvie.com/posts/a-successful-git-branching-model>
License: Creative Commons BY-SA

What you need to know first

- Policies, guidelines and standards at all levels
 - LArSoft design principles
 - Coding guidelines
 - Git branching model
 - Documentation guidelines
- The context for your code
 - Organization of the code



What you need to know first

- Policies, guidelines and standards at all levels
 - LArSoft design principles
 - Coding guidelines
 - Git branching model
 - Documentation guidelines
- The context for your code
 - Organization of the code
 - **art API**

The art wiki

Overview Activity Roadmap Issues New issue Gantt Calendar News Documents Wiki Repository

Documentation [★ Watch](#) [← History](#)

Documentation

General art and art suite information.

- [The art Documentation](#). (This is the *art* SharePoint web site)
- Mailing lists: art-users@fnal.gov (community, also monitored by experts) and artists@fnal.gov (experts).
- [The August, 2015 art/LArSoft course materials](#)
- [Supported Platforms](#)

art suite programs

Each of the programs below is available if you have setup the *art* product (either directly, or by having setup you experiment software that uses *art*). Each program accepts the flag `--help`, which prints out instructions for use.

- **art**: this is the event-processing framework executable. Note the availability of `--debug-config <file>` and `--config-out <file>` to print out the configuration that you have just told *art* to use. Note also `--annotate`, which adds comments that say where each parameter value came from. This command takes into account command-line parameters specified to the *art* executable, which can insert or modify parameters. The *lar* and various experiment-specific framework executables all take these same arguments.
- **art_ut**: this is the event-processing framework executable with built-in support for unit testing using the Boost library. It is intended for writing modules that are used for testing, not for production.
- **config_dumper**: this program will read an *art*/ROOT output file and print out configuration information for the process(es) that created that file.
- **fhicl-dump**: this program will read a FHICL file, fully process it, and print out the FHICL that describes the resulting `ParameterSet`. Note that it requires the environment variable `FHICL_FILE_PATH` to be set. Note that *fhicl-dump* does not take the large set of flags that the framework executable respects.
- **fhicl-expand**: this program will read a FHICL file and perform all `#include` processing, printing the result as a single FHICL document.
- **file_info_dumper** (**new as of 2.01.00**): this program will read an *art*/ROOT output file and has the ability to print the list of events in the file, print the range of events, subruns, and runs that contributed to making the file, and provides access to the internal SQLite database, which can be saved to an external database.
- **count_events**: this program will read an *art*/ROOT output file and print out how many events are contained in that file.
- **product_sizes_dumper**: this program will read and *art*/ROOT output file and print out information about the sizes of products.
- **sam_metadata_dumper**: The *sam_metadata_dumper* application will read an *art*-ROOT format file, and extract the information for possible post-processing and upload to SAM.
- **cetskelgen**. This modular script will generate skeleton source files for the selected module or plugin type. If an experiment designs their own plugin, they can produce a plugin for *cetskelgen* to generate a skeleton for same.

Guides to specific *art* features.

Job configuration.

- [art framework parameters](#).
- [Configuration validation and description](#)
- [FHICL 3 Quick Start Guide](#).
 - [Good art workflow: a presentation](#)

Using and writing *art* modules.

- [art Module Design Guide](#).
- [Product Mixing](#).
- [The ProvenanceDumper output module template](#).

Output-file handling.

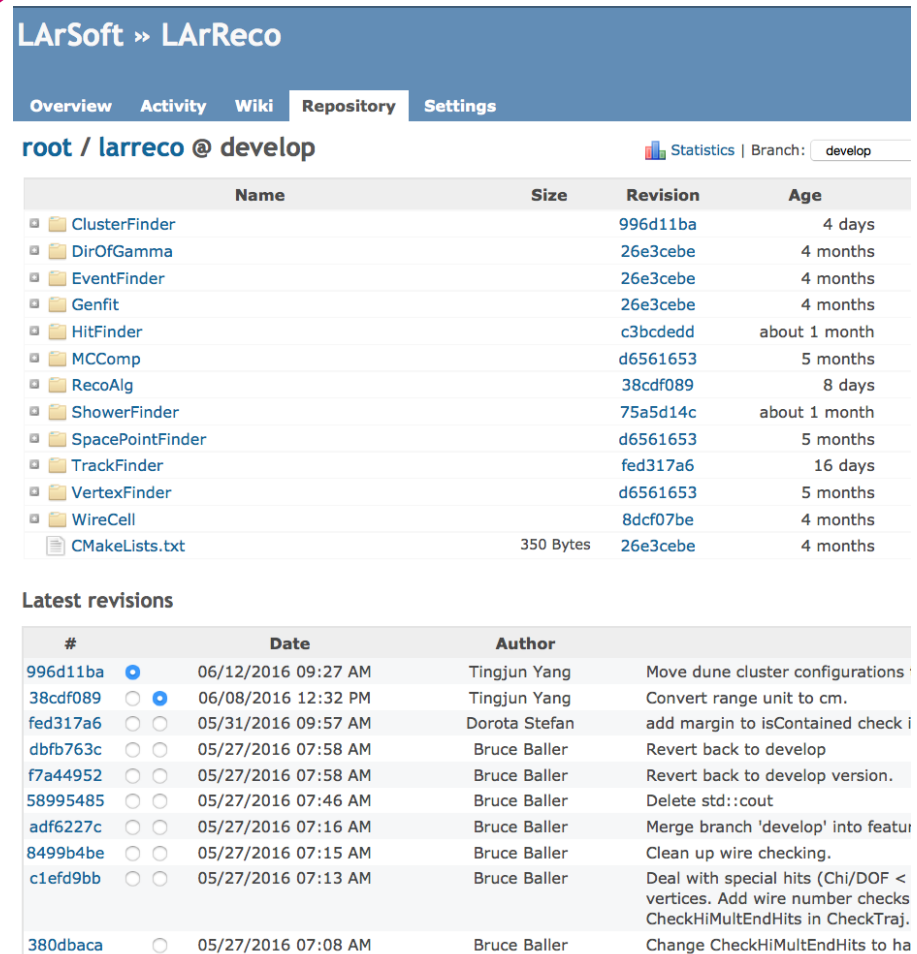
- [art/ROOT output file handling \(art 2.01.00 and newer\)](#)

Documentation
General art and art suite information.
art suite programs
Guides to specific art features.
Job configuration.
Using and writing art modules.
Output-file handling.
Data products and ROOT dictionaries:
Other information and guides for art suite features and packages.
General programming information and advice.
Design documentation.
Art Suite release notes.
Development info.
Meetings
Presentations to Experiments

Also art.fnal.gov

What you need to know first

- Policies, guidelines and standards at all levels
 - LArSoft design principles
 - Coding guidelines
 - Git branching model
 - Documentation guidelines
- The context for your code
 - Organization of the code
 - art API
 - **Relevant existing code**



The screenshot shows the LArSoft LArReco repository page. The top navigation bar includes 'Overview', 'Activity', 'Wiki', 'Repository', and 'Settings'. The current page is 'root / larreco @ develop' with a 'Statistics' link and a 'Branch: develop' dropdown. Below this is a table of files and folders with columns for Name, Size, Revision, and Age.

Name	Size	Revision	Age
ClusterFinder		996d11ba	4 days
DirOfGamma		26e3cebe	4 months
EventFinder		26e3cebe	4 months
Genfit		26e3cebe	4 months
HitFinder		c3bcdedd	about 1 month
MCComp		d6561653	5 months
RecoAlg		38cdf089	8 days
ShowerFinder		75a5d14c	about 1 month
SpacePointFinder		d6561653	5 months
TrackFinder		fed317a6	16 days
VertexFinder		d6561653	5 months
WireCell		8dcf07be	4 months
CMakeLists.txt	350 Bytes	26e3cebe	4 months

Below the file list is a section for 'Latest revisions' with a table showing the commit history.

#	Date	Author	Message
996d11ba	06/12/2016 09:27 AM	Tingjun Yang	Move dune cluster configurations
38cdf089	06/08/2016 12:32 PM	Tingjun Yang	Convert range unit to cm.
fed317a6	05/31/2016 09:57 AM	Dorota Stefan	add margin to isContained check i
dbfb763c	05/27/2016 07:58 AM	Bruce Baller	Revert back to develop
f7a44952	05/27/2016 07:58 AM	Bruce Baller	Revert back to develop version.
58995485	05/27/2016 07:46 AM	Bruce Baller	Delete std::cout
adf6227c	05/27/2016 07:16 AM	Bruce Baller	Merge branch 'develop' into featu
8499b4be	05/27/2016 07:15 AM	Bruce Baller	Clean up wire checking.
c1efd9bb	05/27/2016 07:13 AM	Bruce Baller	Deal with special hits (Chi/DOF < vertices. Add wire number checks CheckHiMultEndHits in CheckTraj.
380dbaca	05/27/2016 07:08 AM	Bruce Baller	Change CheckHiMultEndHits to ha

What you need to know first

- Policies, guidelines and standards at all levels
 - LArSoft design principles
 - Coding guidelines
 - Git branching model
 - Documentation guidelines
- The context for your code
 - Organization of the code
 - art API
 - Relevant existing code
- Tools
 - **mrbs, cmake, cetbuildtools, debuggers, profilers, CI system...**



The point of this model

- Focus is on
 - producing shareable, relatively uniform code
 - maintaining a stable development environment
 - This is balanced with recency (i.e., proximity to the head of develop branch)
 - finding consensus across experiments for changes
 - managing integration and deployment of changes

In short: on **gate-keeping**

- Good for users, but takes from agility, usability for developers
 - Steep learning curve
 - Multiple barriers to rapid development
 - Haven't even discussed other aspects of development environment, eg, built speed.

Finding the right balance

- Want to find the sweet spot for the general case
 - Enough agility to keep people interested in producing code
 - Enough gate-keeping to keep people using it
- Are we in the right spot?
 - Meetings?
 - Important to discuss changes before they happen
 - Also facilitate discussions about
 - improvements, other changes affecting lots of people
 - priorities, policies, procedures, etc.
 - But is the present requirement reasonable?
 - Standards and policies?
 - Would structuring releases differently help?
 - Would structuring code differently help?
 - Changes to development environment that might help?



Discuss amongst ourselves...



...then coffee break



The end