Architecture review status report

Core service review

Gianluca Petrillo

Fermi National Accelerator Laboratory

LArSoft Coordinators' Meeting, November 17th , 2015

Fermilab

Long term plan

This is old information, still valid...

- this review project will be *terminated at the end of the year*
- planned deliverables are shown in the wiki page
  - factorization of core service
  - a small selected number of reconstruction modules

  considered how much time I have available, it's unlikely for any of the optional modules to be delivered...
- a second phase will follow, where the requirements from the LArTPC workshop will be implemented

Status: core services

Core services have been fully refurbished by Jonathan Paley and me:

- factorization
- abstraction
- renaming
- rearranging

All these are breaking changes...

Status: core services renaming and rearranging

Core services *(providers have same name minus* `Service`*)*:

`LArPropertiesService`	
`DetectorPropertiesService`	
`DetectorClocksService`	(was: `TimeService`)
`DatabaseUtil`	*being dismantled*

Rearrangement:

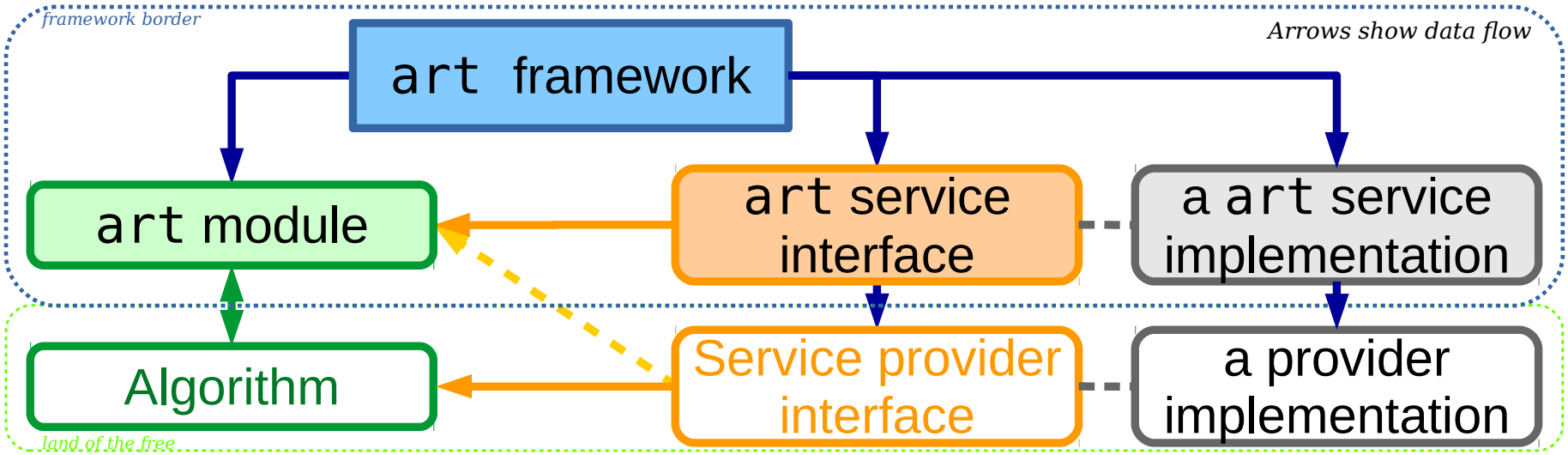
`Efield()`	`LArProperties` →`DetectorProperties`
`DriftVelocity()`	`LArProperties` →`DetectorProperties`
`ElectronLifetime()`	`LArProperties` →`DetectorProperties`
`Temperature()`	`LArProperties`... still there so far

Details in LArSoft's core service review wiki page.

Status: core services factorization and abstraction

Core services now follow the factorization and abstraction model:

- framework-independent code can use them
- an experiment can define its own implementation (or use a “standard” one)



Unit tests should be easier now...

How to fix the breakage

Details on how to update your code are at
https://cdcvs.fnal.gov/redmine/projects/larsoft/wiki/Core_Services_Review.
Changes include fixing:

- class names and service access (easy: you don't, compiler complains)
- calls to moved methods (easy, as above)
- linkage (mostly easy: just need to remove stuff)
- FHiCL configuration (harder: you might not notice until runtime)

Update status:

LArSoft	fully updated
MicroBooNE	fully updated (needed workarounds)
DUNE	updated by Jonathan
others	(LArIAT, SBND, ArgoNeuT) on my list

If you have code off develop, you'll have to update it yourself
⇒ feel free to ask me for support!

To-do list

A number of things are still to be done:

- set up legacy code for ArgoNeuT (critical)
- rename channel status and pedestal services to follow the established pattern (*way* better be soon)
- move `Temperature()` to `DetectorProperties` (better be soon)
- remove `DatabaseUtil` from where it's not needed (can wait)
- verify and update all `art::ServiceHandle` uses and service dependences (can wait)
- fix the design issue with `DetectorClocksStandard` (can wait)
- write unit tests (can wait)

Some of these won't make it for the merging and will be postponed.

Short-term plans

- LArSoft work completed, unit tests run
- MicroBooNE code updated, unit tests run, C.I. tests not run
- service refactoring should be complete by the next coordination meeting
- I'll ask the final word for merging at that time
  (I'll accept the final word now if you feel ready)
- then algorithms factorization as planned
- MicroBooNE code is ready for extensive testing!!
- code (updated to LArSoft `v04_29_01`):

  LArSoft `feature/jpaley_LArPropertiesBreakup`
  `uboonecode` `feature/gp_ServiceCoreReview`

Backup

Dismantling `DatabaseUtil`

`DatabaseUtil` service contains two classes of methods:

1. direct access to a database, only used by ArgoNeuT
2. MicroBooNE specific database access code

What's going to happen:

1. a legacy version of core properties will be created for use to ArgoNeuT
2. MicroBooNE specific code will be moved to `uboonecode`
3. each service in need will implement access to a specific DB
4. recommended access is not direct but via `libwda`
5. if a common pattern will emerge, common functions can be collected and coagulated into a service

`DetectorClocksStandard` design issue

- core service providers assume they are always up to date
- users should also assume that
- the framework is responsible to update them as needed
- `DetectorClocksStandard` providers information based on a data product (`raw::Trigger` collection)
- that data product might have been created in this job, and `DetectorClocksStandard` would not know
- MicroBooNE code explicitly forced `TimeService` to check
- such a command is not available in `DetectorClocks` interface

There are a few possible solutions, depending on complexity.