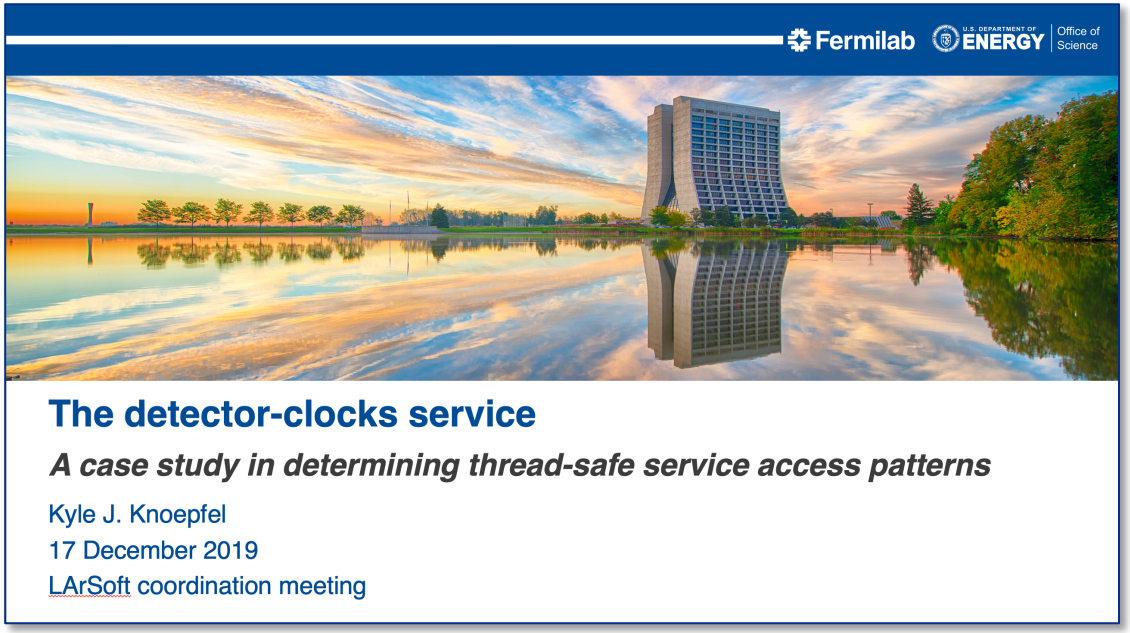
Required changes for thread safety and some observations

Kyle J. Knoepfel

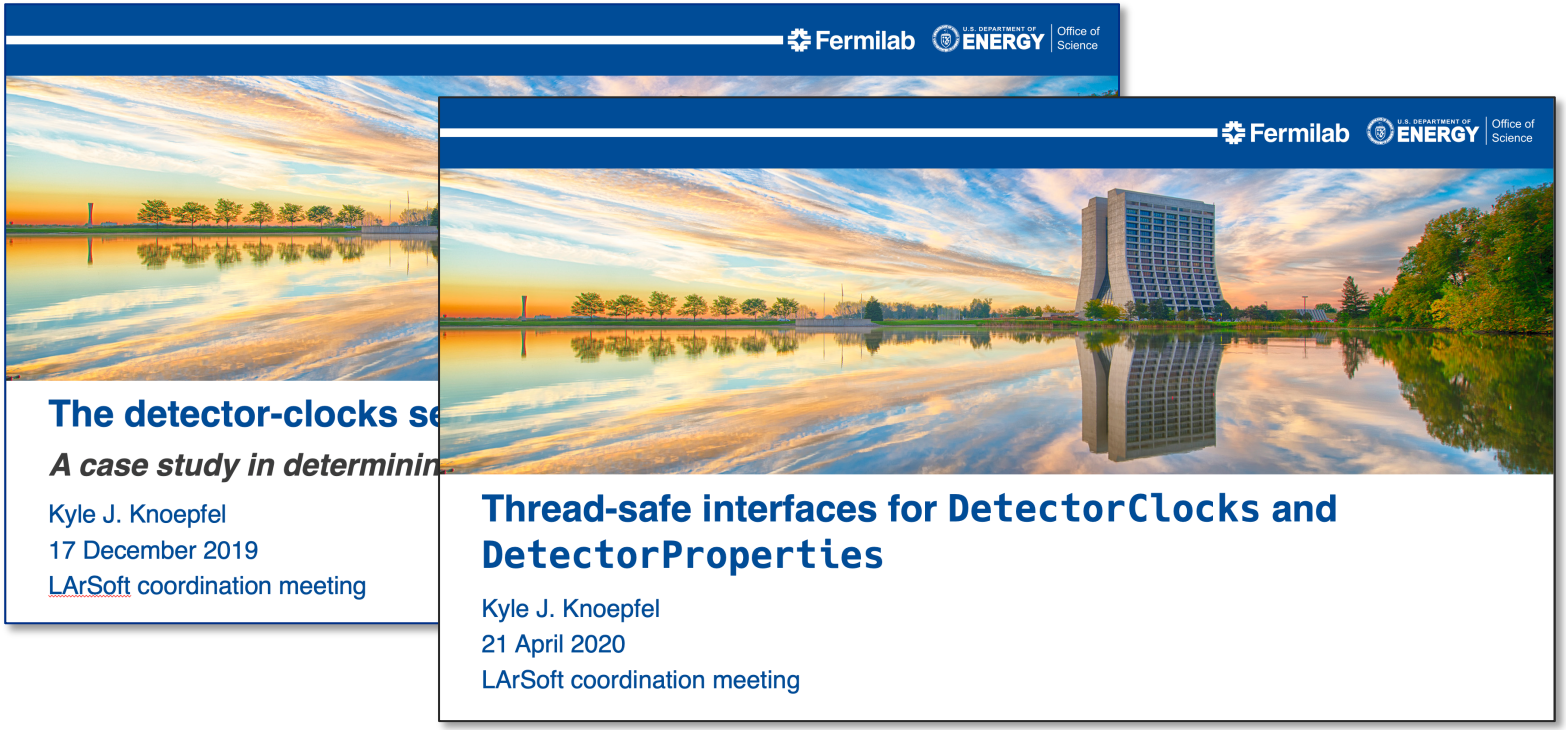
19 May 2020

LArSoft coordination meeting

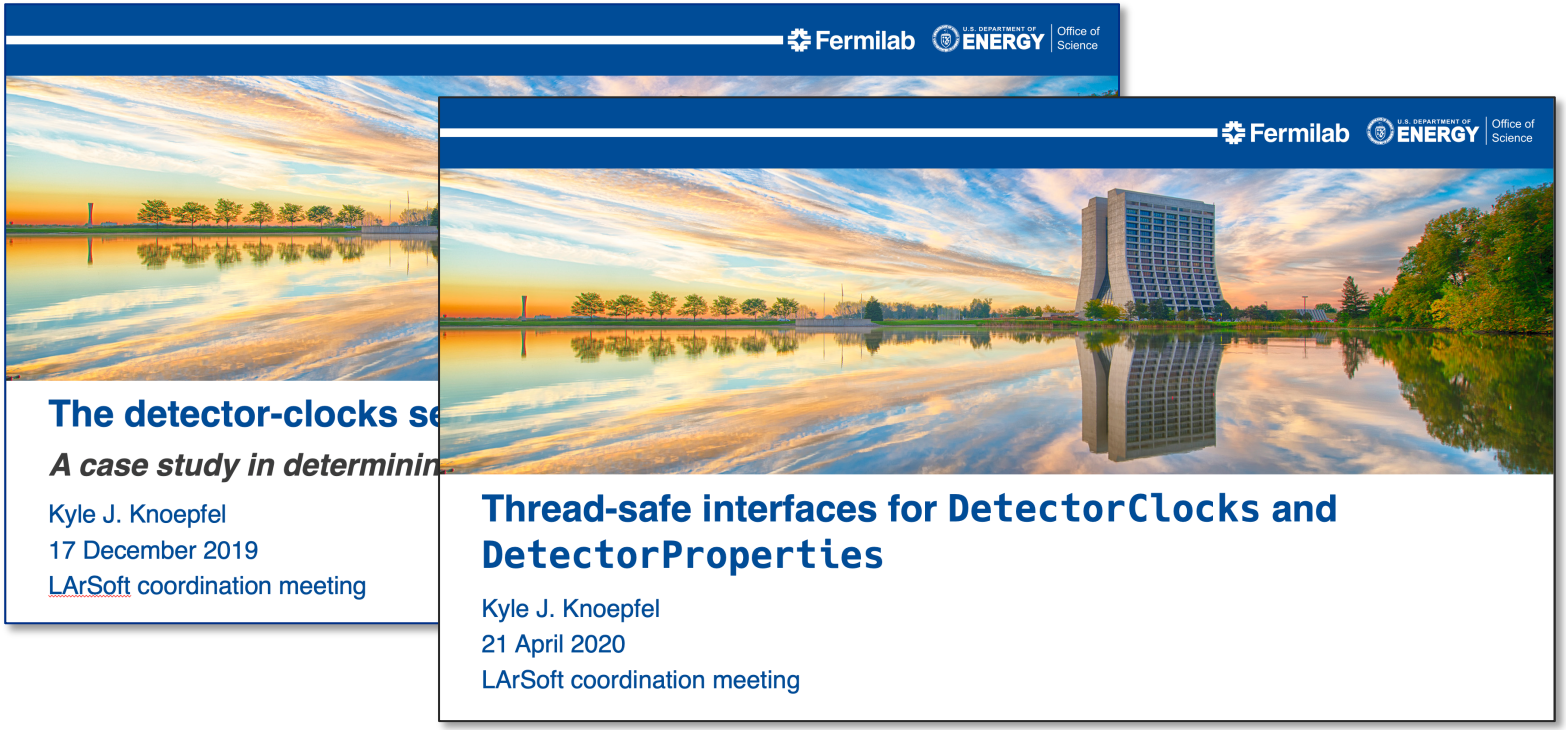
From previous meetings



From previous meetings



From previous meetings



This talk presents final details

Proposal from April 21

- **Proposal:** The “persistent data structures” approach should be adopted for the `DetectorClocks` and `DetectorProperties` providers and services.

- Data structures are constant upon creation; adjusting the data structure dynamically occurs by layering newly created objects upon extant ones.

- This proposal agreed upon at the last coordination meeting.

Fermilab

What does `DetectorClocks` look like?

- As only events within a subrun can be processed concurrently at the moment, only event-level data must be thread-safe.
- The majority of the `DetectorClocks` interface still exists, but there is an extra layer in between called `DetectorClocksData`.

Fermilab

What does `DetectorClocks` look like?

- As only events within a subrun can be processed concurrently at the moment, only event-level data must be thread-safe.
- The majority of the `DetectorClocks` interface still exists, but there is an extra layer in between called `DetectorClocksData`.

Old interface

```
using detinfo::DetectorClocksService;

MyProducer::MyProducer(ParameterSet const& pset)
{
  ServiceHandle<DetectorClocksService const> clocks;
  double beam_time = clocks->BeamGateTime();
}


void MyProducer::produce(art::Event& e)
{
  ServiceHandle<DetectorClocksService const> clocks;
  double beam_time = clocks->BeamGateTime();
}
```

New interface

```
using detinfo::DetectorClocksService;

MyProducer::MyProducer(ParameterSet const& pset)
{
  ServiceHandle<DetectorClocksService const> clocks;
  auto const clockData = clocks->DataForJob();
  double beam_time = clockData.BeamGateTime();
}

void MyProducer::produce(art::Event& e)
{
  ServiceHandle<DetectorClocksService const> clocks;
  auto const clockData = clocks->DataFor(e);
  double beam_time = clockData.BeamGateTime();
}
```


Fermilab

What does `DetectorProperties` look like?

- As only events within a subrun can be processed concurrently at the moment, only event-level data must be protected.
- The majority of the `DetectorProperties` interface still exists, but there is an extra layer in between called `DetectorPropertiesData`.

Old interface

```
using detinfo::DetectorPropertiesService;

MyProducer::MyProducer(ParameterSet const& pset)
{
  ServiceHandle<DetectorPropertiesService> detProp;
  double dv = detProp->DriftVelocity(…);
}


void MyProducer::produce(art::Event& e)
{
  ServiceHandle<DetectorPropertiesService> detProp;
  double dv = detProp->DriftVelocity(…);
}
```

New interface

```
using detinfo::DetectorPropertiesService;

MyProducer::MyProducer(ParameterSet const& pset)
{
  ServiceHandle<DetectorPropertiesService> detProp;
  auto const dPropData = detProp->DataForJob();
  double dv = dPropData.DriftVelocity(…);
}

void MyProducer::produce(art::Event& e)
{
  ServiceHandle<DetectorPropertiesService> detProp;
  auto const dPropData = detProp->DataFor(e);
  double dv = dPropData.DriftVelocity(…);
}
```


Fermilab

DetectorProperties under the covers…

- Some aspects of DetectorProperties relies on DetectorClocks.
- Although this is implicit, if you are using DetectorClocks and DetectorProperties in the same scope, you can optimize slightly.

```
using DetClocks = ServiceHandle<detinfo::DetectorClocksService>;
using DetProps = ServiceHandle<detinfo::DetectorPropertiesService>;

MyProducer::MyProducer(ParameterSet const&)
{
  auto const clockData = DetClocks{}->DataForJob();

  auto const dPropData1 = DetProps{}->DataForJob();       // Internally creates 'clockData' object
  auto const dPropData2 = DetProps{}->DataFor(clockData); // More efficient
}

void MyProducer::produce(art::Event& e)
{
  auto const clockData = DetClocks{}->DataFor(e);

  auto const dPropData1 = DetProps{}->DataFor(e);         // Internally creates 'clockData' object
  auto const dPropData2 = DetProps{}->DataFor(clockData); // More efficient
}
```


Fermilab

Consequences of this design

- Configurations can remain the same.
- Code using the `DetectorClocks/Properties` services must know if event-level data or global data is needed.
- Framework-agnostic code that creates a `DetectorClocks/Properties` service handle must be adjusted to receive the correct information.
  - Can no longer access the provider directly through service.

```
MyProducer::MyProducer(ParameterSet const& pset)
{
  auto clocks = lar::providerFrom<detinfo::DetectorClocksService>();    // ! Compile-time error
  auto dprop = lar::providerFrom<detinfo::DetectorPropertiesService>(); // ! " "
}
```

- The provider/service hierarchies still exist, and the providers are usable with gallery programs, albeit with a different interface.

Fermilab

Other breaking changes

- In short: "**remove as much mutable state as possible**"
  - Many 'reconfigure' functions in larreco have been removed.
  - `detinfo::ElecClock` interface has changed
  - Many `BackTracker` functions now require a `DetectorClocksData` argument.
  - Various `larreco` class interfaces now take a `DetectorClocksData` and/or `DetectorPropertiesData` argument.
- *Tangential:* `boost::optional` use has been replaced with `std::optional`
- Any code using `DetectorPropertiesData` must link against `lardataalg_DetectorInfo`.
- ArgoNeuT's detector-properties service in `lardata/Utilities` has been refactored into a provider and a service.
  - First step toward moving the ArgoNeuT-specific services from lardata to argoneutcode.

Fermilab

LegacyLArG4

- Custom physics lists in larsim are currently enabled through the use of static variables constructed whenever `liblarsim_LegacyLArG4.so` has been loaded.
- This is a weak design as one must be careful when the library is loaded.
- With the adjusted interface, LArSoft's custom physics lists are no longer created automatically at library load time.
  - An explicit call must be made in the constructor of (e.g.) LArG4 to load them:

```
LArG4::LArG4(fhicl::ParameterSet const& pset)
  : art::EDProducer{pset}
  , …
  , fAllPhysicsLists{art::ServiceHandle<DetectorPropertiesService>()->DataForJob()}
{
   MF_LOG_DEBUG("LArG4") << "Debug: LArG4()";
   …
}
```


Fermilab

Disentangling detector clocks/properties

- `DetectorProperties` has provided sampling-rate and trigger-offset accessors.
  - But those functions rely only on the clocks service.

Replace:	`auto detprop = lar::providerFrom<detinfo::DetectorPropertiesService>();`
	`auto rate = detprop->SamplingRate();`
	`auto offset = detprop->TriggerOffset();`

With:	`auto clockData =`
	`  art::ServiceHandle<detinfo::Detector**Clocks**Service>()->DataFor(e);`
	`auto rate = **sampling_rate**(clockData);`
	`auto offset = **trigger_offset**(clockData);`

Fermilab

Disentangling detector clocks/properties

- `DetectorProperties` has provided sampling-rate and trigger-offset accessors.
  - But those functions rely only on the clocks service.

Replace:	`auto detprop = lar::providerFrom<detinfo::DetectorPropertiesService>();`
	`auto rate = detprop->SamplingRate();`
	`auto offset = detprop->TriggerOffset();`

With:	`auto clockData =`
	`  art::ServiceHandle<detinfo::DetectorClocksService>()->DataFor(e);`
	`auto rate = sampling_rate(clockData);`
	`auto offset = trigger_offset(clockData);`

- I see a lot of places like this:

```
double timetick = sampling_rate(clockData) * 1e-3; // time sample in us
```

- In the future, I suggest an interface like:

```
double timetick = sampling_rate<microseconds>(clockData);
```


Fermilab

Removed files

- The following files had dependencies on detector clocks and properties, but they were not used anywhere in the LArSoft code base or in experiment's develop branches.

larreco/Calorimetry/
- LinearEnergyAlg.*

larreco/RecoAlg/CMTool/CMTAlgMatch/
- CFAlgo3DAngle.*
- CFAlgoChargeDistrib.h
- CFAlgoStartPointCompat.*
- CFAlgoStartPointMatch.*
- CFAlgoStartTimeCompat.*
- CFAlgoVolumeOverlap.*
- CFAlgoWireOverlap.*
- *CMTAlgMatch-TypeDef.h*
- *LinkDef.h*
- *bin*

larreco/RecoAlg/CMTool/CMTAlgMerge/
- CBAlgoShortestDist.*
- CBAlgoShortestDistNonEndPoint.*
- CBAlgoShortestDistSmallCluster.*
- CBAlgoStartInCone.*
- CBAlgoStartInPoly.*
- CBAlgoTrackSeparate.*
- *CMTAlgMerge-TypeDef.h*
- *LinkDef.h*
- *bin*

larreco/RecoAlg/CMTool/CMTAlgPriority/
- *CMTAlgPriority-TypeDef.h*
- *LinkDef.h*
- *bin/*
- *lib/*

larreco/RecoAlg/CMTool/CMToolBase/
- *CMTool-TypeDef.h*
- *LinkDef.h*
- *lib/*

larreco/RecoAlg/CMTool/mac

- If this is a problem, let me know.

Fermilab

Observations in experiment code

Fermilab

Unnecessary typing

- There were many places where detector-clocks/properties pointers were being cached but not used.
  - This creates a maintainability problem, and a lot of extra time porting code.
  - **If you're not using a variable, please remove it.**

- There were many places where code was repeated from one module to another.
  - Cut-and-paste is easy upfront, but it is a maintenance nightmare.
  - Each place in the code has to be updated whenever the interface changes.
  - **Refactor the code so that you "don't repeat yourself" (DRY).**

- **If you want multi-threaded code sooner, please remove duplicate code.**

Fermilab

Numerics

- Many places where code could be clearer, more concise, and more efficient:

Replace …	with …
`TMath::Xyz(…)` *[e.g. `TMath::Sqrt(…)`]*	`std::xyz(…)` *[e.g. `std::sqrt(…)`]*
`std::pow(x, 0.5)`	`std::sqrt(x)`
`std::pow(x, 2)`	`cet::square(x)`
`std::sqrt(std::pow(x, 2) + std::pow(y, 2))`	`std::hypot(x, y)`

Fermilab

Miscellany

- **Passing by value vs. reference**
  - Understand in which contexts it makes sense to pass by value, and when it makes sense to pass by reference.
  - My general guidance:

```
if size of data == 8 to 24 bytes:
  pass by value
else
  pass by reference
```


Fermilab

Miscellany

- **Passing by value vs. reference**
  - Understand in which contexts it makes sense to pass by value, and when it makes sense to pass by reference.
  - My general guidance:

```
if size of data == 8 to 24 bytes:
  pass by value
else
  pass by reference
```

- **Highly nested logic**
  - If there is a lot of whitespace to the left of your code, then you probably need to refactor the code.

Fermilab

Misc

- Pas
  - U
    to
  - M
- High
  - If
    th

sense

ctor

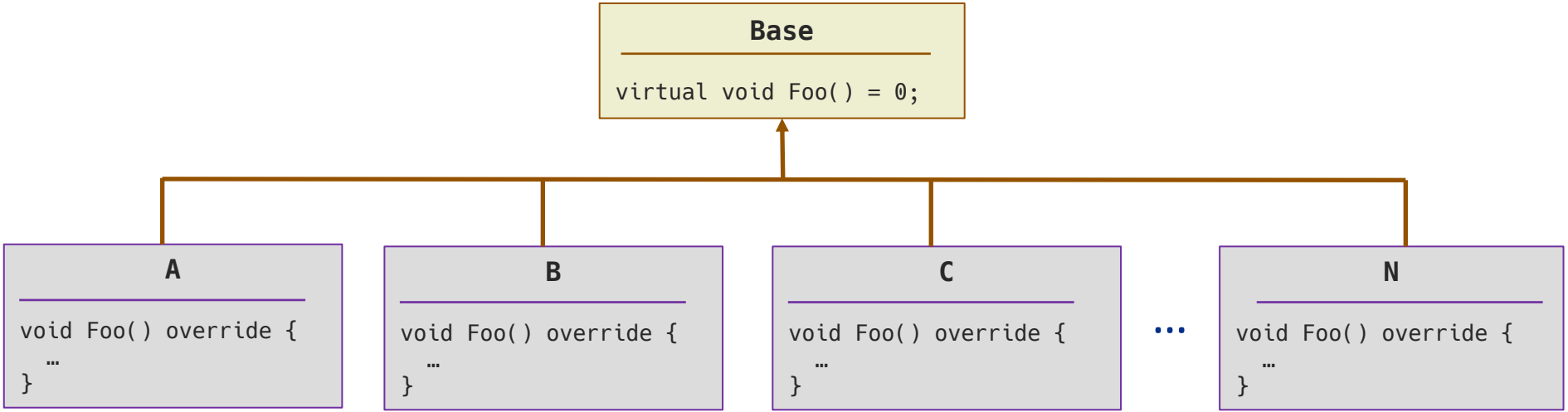
```
                                                true_mu_median_len_ratio->Fill((float(med_ftrstmu-med_laststmu))/mud_tlen);
                                                true_all_trk_median_dedx->Fill(med_allk);
                                                true_all_trmu_median_dedx->Fill(med_allmu);
                                              } // trk g4 match
                                            } // getting true mu
                                          } // getting scttering particles
                                        } // 2 nd loop over geant list for P
                                      } // getting stopping K
                                    } // getting primary K...
                                  } // 1 st loop over geant list for P...
                                } // break indicator...
                                //} // Mu mom cut ....
                                //} // Med all mu dE/dx...
                              } // Mu PIDA .....
                              //} // Med all K dE/dx...
                            } // K PIDA cut ......
                            //} // K/Mu min-max hit cut .....
                            //} // Muon tracklength cut ......
                          } // K/Mu open angle cut .....
                        } // K/Mu energy gradient cut.....
                      } // decay mu connected to start of K
                    } // primary mu is connected to end of K
                    ////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

                      } // decay mu is contained...
                    } // more than 10 hits for dec mu....
                    //} // 3rd loop over ical for i_2 track....
                  } // trklen cut for decaying mu
                } // 3 rd loop over tracks
              } // primay K and primay Mu distance matching....
            } // pri K contained...
          } // more than 10 hits for Pri K...
          //} // 2 nd loop over ical for i_1 track....
        } // pri K >= 10 cm
      } // 2 nd loop over tracks...
    } // more thans 10 hits for pri mu...
    //} // 1 st loop over cal of i track...
    //} // calorimetry info saved...
  } // primary mu >= 60 mc ...
} // 1 st loop over tracks....
} // tracks > 3 in the event....
} // track infomation is saved.....
}// is not data (is MC) if statement.........
```

Time to refactor…

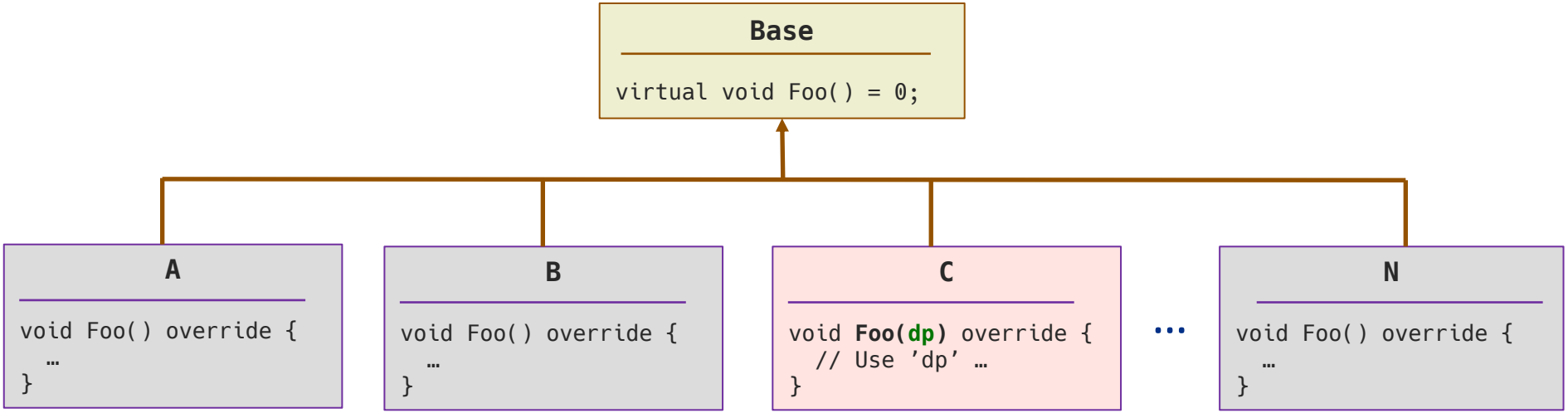
Fermilab

Challenges with inheritance



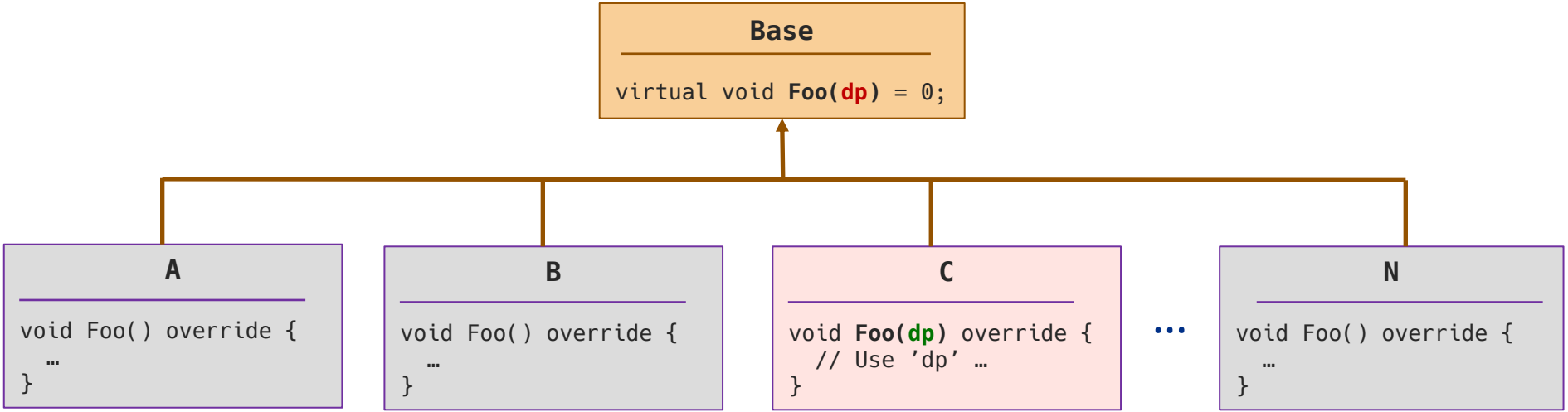
Fermilab

Challenges with inheritance



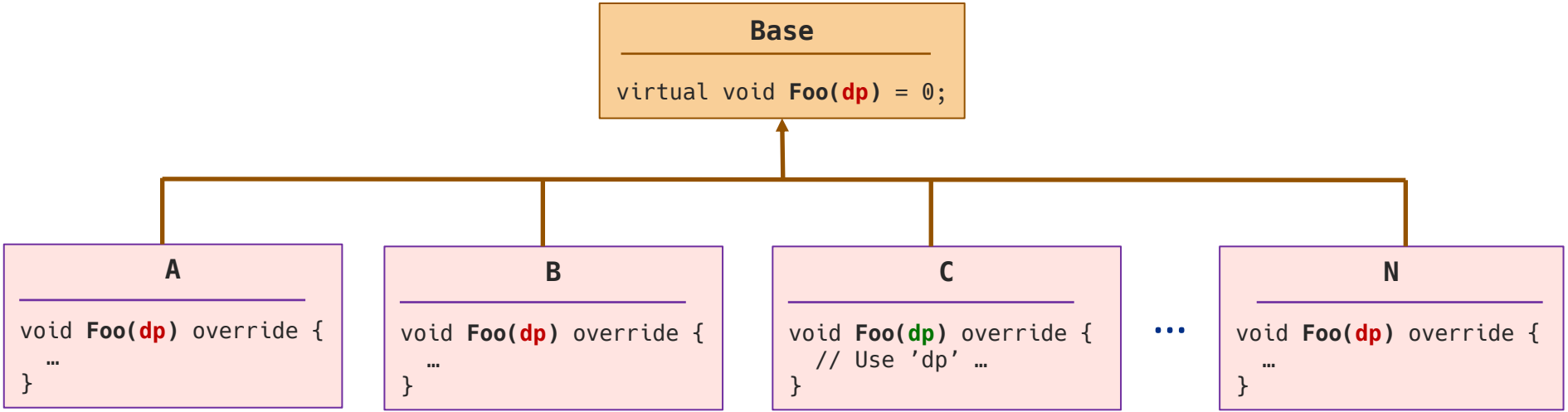
Fermilab

Challenges with inheritance



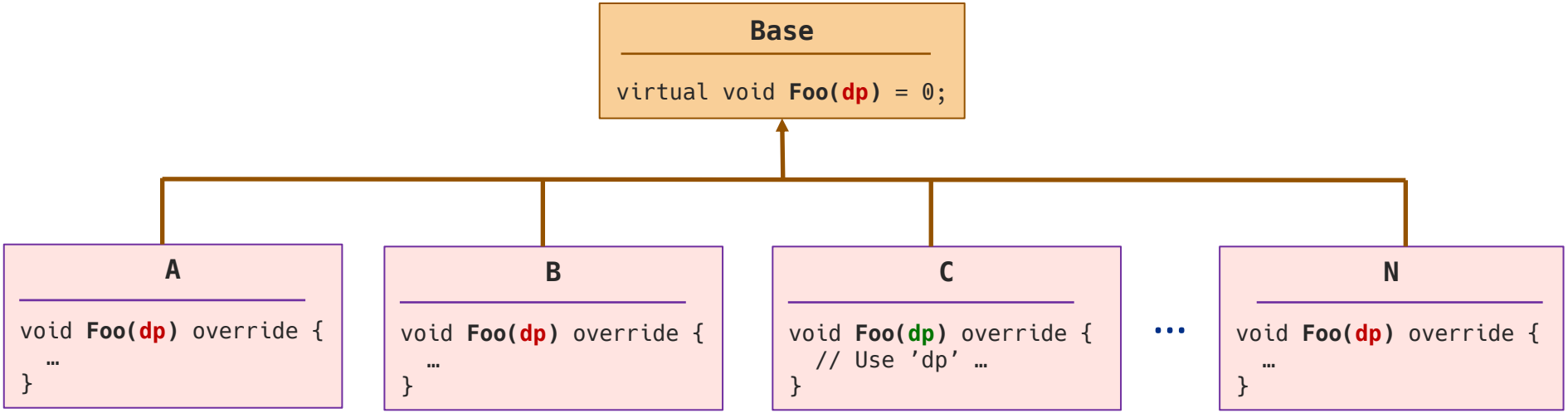
Fermilab

Challenges with inheritance



Fermilab

Challenges with inheritance



- If only one implementation requires an interface change, all are affected.
- Before you implement and start using an interface:
  - Make sure it is the best solution for the task to be accomplished.
  - Expect to have to adjust the interface over time.

Fermilab

Library dependencies

- Some products support target-based CMake library specifications:
  - **ROOT**: Replace (e.g.) `${ROOT_CORE}` with `ROOT::Core`
  - **Boost**: Replace (e.g.) `${Boost_FILESYSTEM_LIBRARY}` with `Boost::filesystem`
- Remove the `Boost::system` (or `${Boost_SYSTEM_LIBRARY}`) dependency from your CMakeLists.txt files.
- All modules or services compiled with `simple_plugin` or `art_make` have some basic libraries included automatically.

Modules	`art_Framework_{Core,Principal}, art_Persistency_{Common,Provenance}, art_Utilities`
	`canvas, fhiclcpp, cetlib, cetlib_except,`
	`Boost::filesystem`

Services	`art_Framework_Services_Registry, art_Persistency_Common, art_Utilities`
	`canvas, fhiclcpp, cetlib, cetlib_except,`
	`Boost::filesystem`

Fermilab

Conclusions

- Aim to have PRs and experiment feature branches ready by tomorrow (Wednesday) morning.
- Feature branch names will be `feature/knoepfel_detector_services`
- Please look at the feature branches to determine how some of the experiment-specific interface has changed.
  - If you have concerns, please contact me.
- Thank you for your patience.
  - This has been a long slog, but many LArSoft underpinnings will either be thread-safe or be easier to make thread-safe.

Fermilab