SoLID Software Framework Selection

Ole Hansen

Jefferson Lab

February 4, 2016

Our High-Level Requirements

- Consistency: Framework should support all of simulation, digitization, reconstruction and physics analysis
- Multi-pass processing: output $\rightarrow$ input for next pass
- Run-time configurable:
  - No recompilation for different analysis workflows/parameters
  - Multiple instances of modules (with different configurations)
- Multiple analysis chains per job, *e.g.*
  - Different tracking or PID schemes
  - Several physics analyses in parallel
- Digitization able to write raw data format
- Extensive metadata in DSTs, *e.g.*
  - Database parameters from previous stages (geometry etc.)
  - Data provenance
- Interactive analysis with ROOT

Apparent State Of the Art Architecture

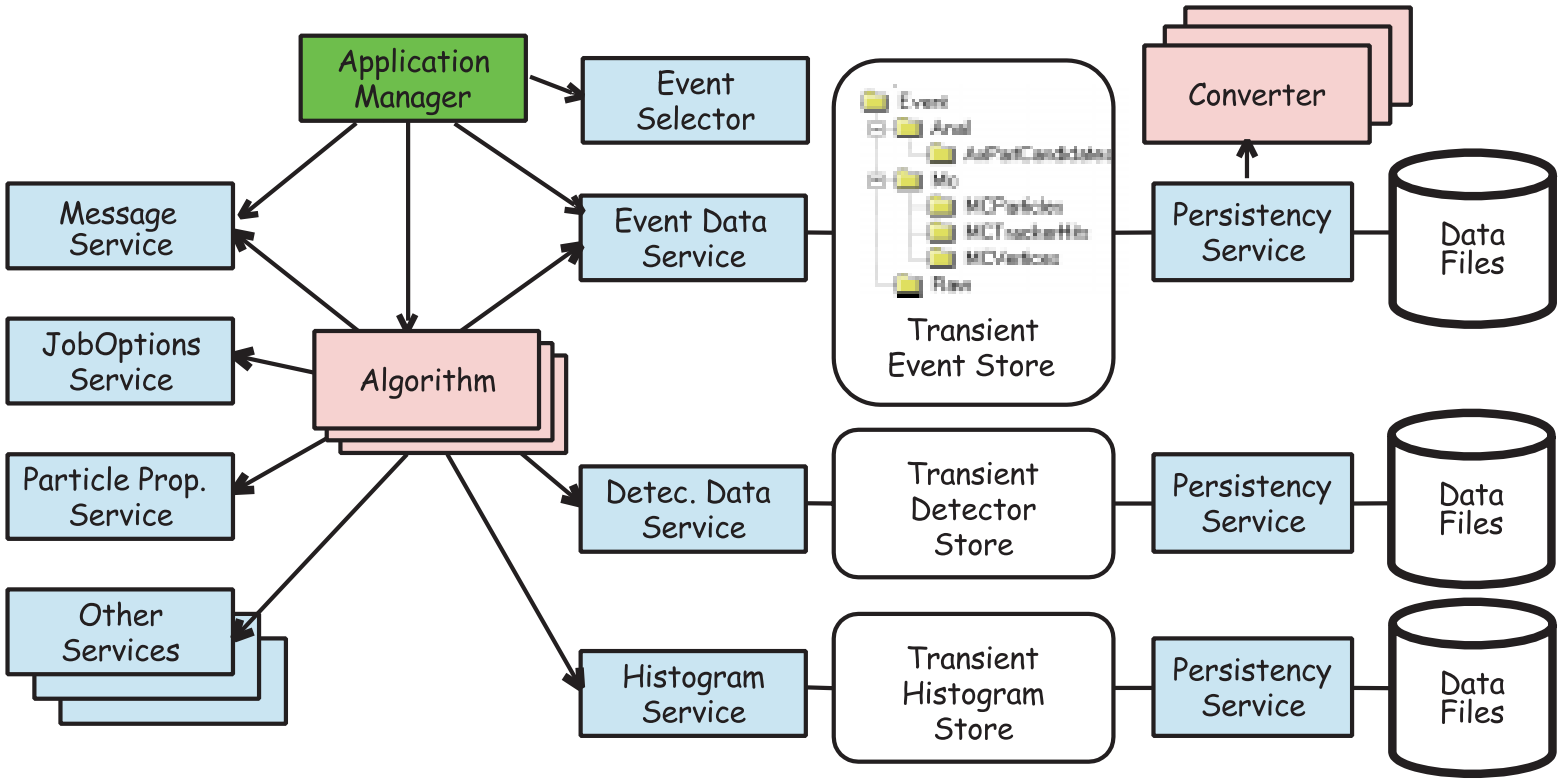
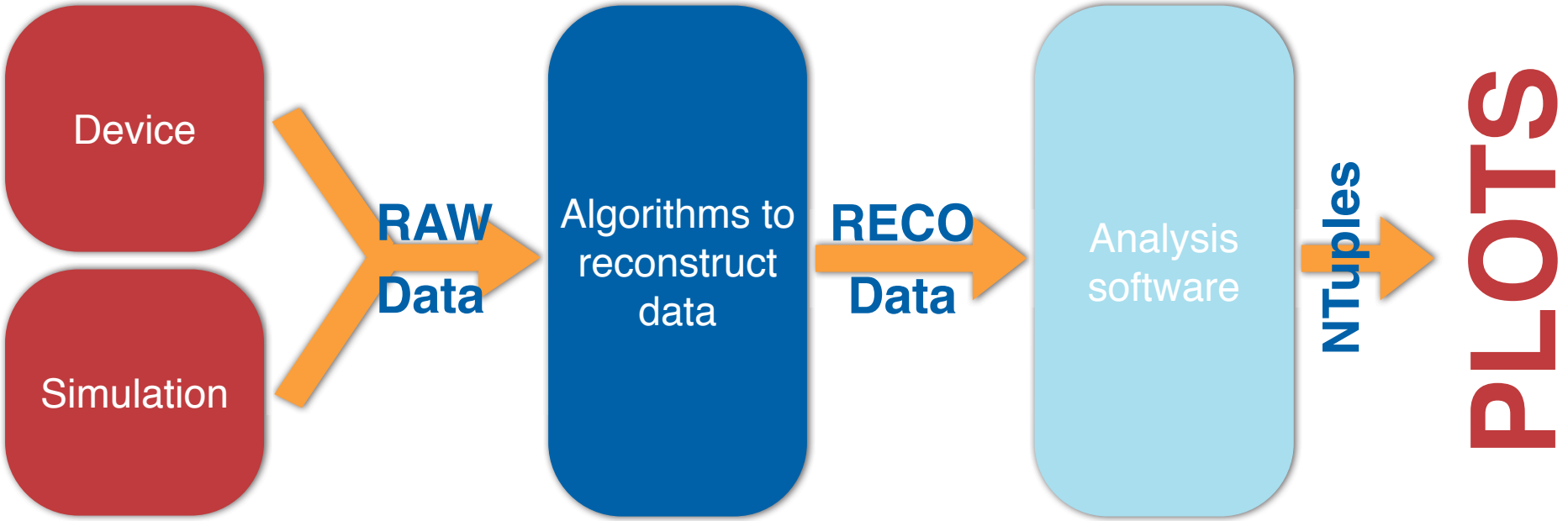


Figure 2: Object Diagram of the GAUDI Architecture

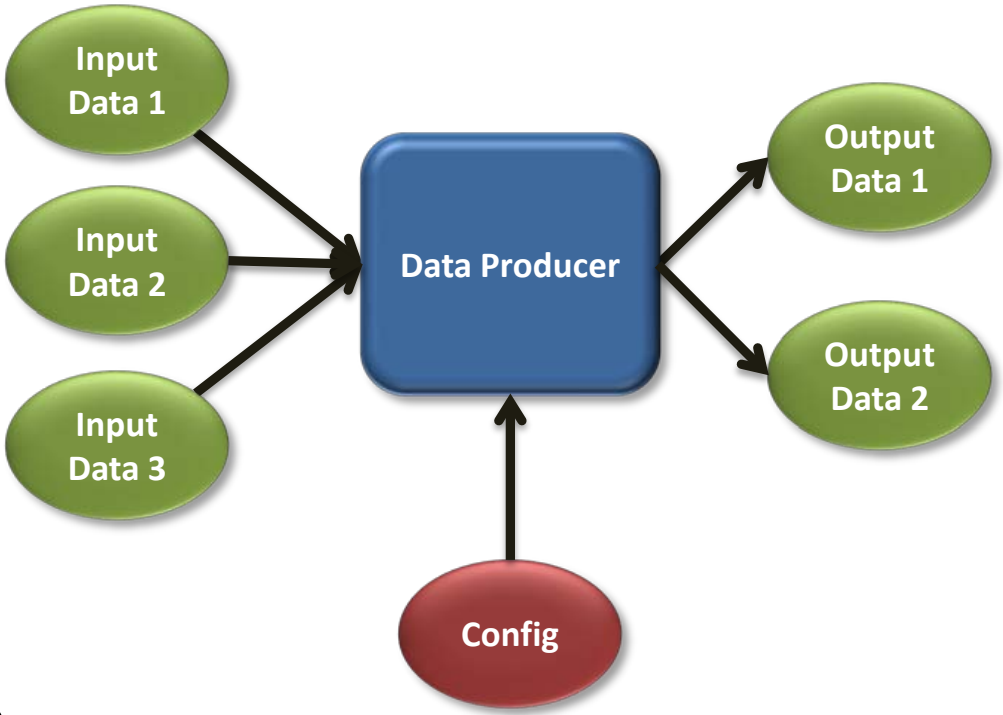
From G. Barrand *et al.*, "GAUDI - A software architecture and framework for building LHCb data processing applications", CHEP2000

Data Flow Cartoon



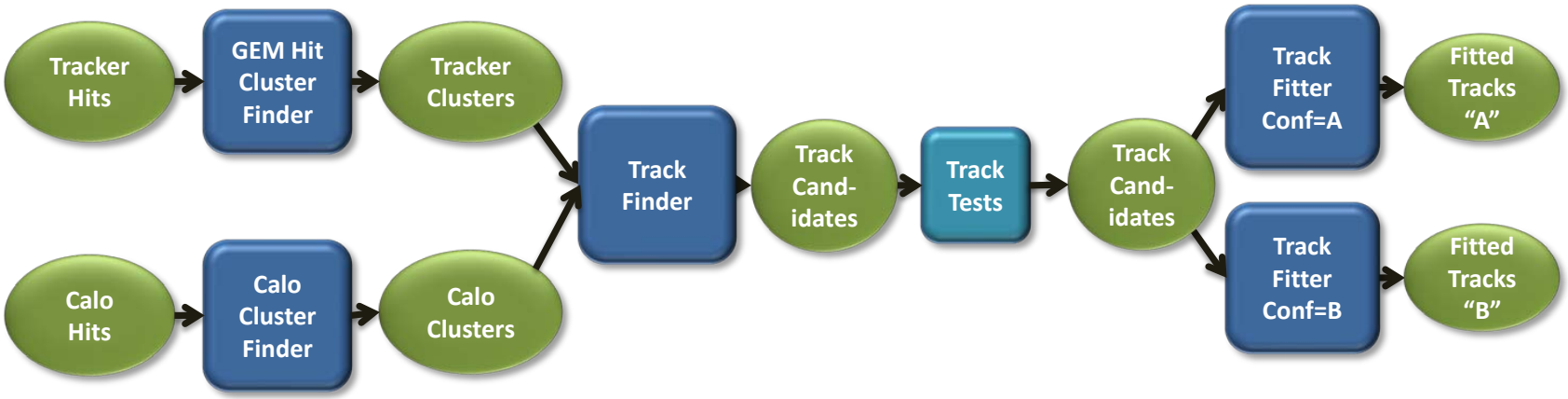
From O. Gutsche (FNAL), "Future Directions in HEP Software and Computing", DPF2015

Algorithms & Data Objects



- Data producers (algorithms)
  - Run-time configurable → reusable w/o recompilation, multiple instances
  - Single algorithm per module → testable independent of framework (unit tests)
- Data objects (inputs & results)
  - separate from producers → decouples algorithms
  - may reference other data objects
  - hold metadata

Analysis Chains



- Modules communicate exclusively via data objects
- Module relationships configurable at run time by selecting from available compatible input data objects (by name, class, instance or similar)
- Condition testing modules select subset of results and/or skip further processing if certain tests fail or succeed.
- Multiple chains per job
- One or more output modules (not shown) write user-configured subset of available data objects and/or ntuples/histograms

LHC Frameworks

- CMS: CMSSW
- LHCb: Gaudi
- ATLAS: Athena (based on Gaudi)
- ALICE: AliRoot

Extremely complex software, total overkill for SoLID

Lighter-Weight Frameworks Comparison

Feature	art (FNAL)	FairRoot (GSI)	Fun4All (PHENIX)	JANA (JLab)
Origin	CMSSW (CMS)	AliRoot (ALICE)	In-house	In-house
First release	2009	2005	2003	2005
Collaborations using framework	~9	~10	1	1
Language	C++11/14	ROOT C++ (pre STL)	ROOT C++ (pre STL)	C++98
Base framework	self-contained	ROOT	ROOT	self-contained
Output, object persistency	custom ROOT	plain ROOT	custom ROOT	HDDM (XML)
Steering, configuration	FHiCL	ROOT macro	ROOT macro	command line & compiled in
Reusable/multi-instance modules	yes	(user)	(user?)	very limited
Multiple analysis chains	yes	yes	yes	very limited
Data product identification	type + 3 keys	type + producer	name	type + tag
Complexity of data object search	O(logN)	O(1)	O(N)	O(M>N)
Data provenance tracking	yes	no	no	no
Test/filter modules to skip event	yes	output module	output module	output module
Thread-safe code	yes	no	no	yes (partial)
Main dependencies	cet-is (3.5 GB)	FairSoft (2.8 GB)	ROOT, boost (1 GB)	Xerces XML
Preferred installation	Binary via UPD	Source (GitHub)	Source (GitHub)	Source (GitHub)
Unit tests	425	39 (high-level)	0	0
User documentation	User Guide (500p), workshops	Examples, Wiki	Examples, Wiki	Examples, Wiki, User Guide (old)
User code reusable for SoLID	some (DB, I/O)	much (Panda, EIC)	some (PHENIX)	much (GlueX)

Lighter-Weight Frameworks Services Features

Feature	art (FNAL)	FairRoot (GSI)	Fun4All (PHENIX)	JANA (JLab)
Transient event store	Event, run, subrun objects	ROOT folders	Phool Node Tree	With producers
Persistency Service	custom ROOT I/O	plain ROOT I/O	custom ROOT I/O	(not part of JANA)
Folders in event store	no	yes	yes	no
Event Data Service	template function	TClonesArray	template function	template function
Message service	yes	yes	no	yes
JobOptions Service	FHiCL API	FairRuntimeDb	no	ParameterManager
Geant4 integration	artG4	VMC	yes (?)	no
Detector Data Service (geo)	no (service API)	no (geo classes)	no	JGeometryXML
Detector Data Service (cond)	no (service API)	no	no	JCalibrationCCDB
Histogram Service	TFileService	no	HistoManager	no
Interactive mode	no	yes	yes	no
Configuration test	yes	no	no	no
Memory tracker	yes	ROOT memcheck	no	no
Polymorphic data objects	yes	yes	no (?)	yes
Inter-object references	art::Ptr, art::Assns (1-1, 1-N, N-N)	TRef, FairLink (?)	integer indices (?)	integer indices

Example FairRoot/EICRoot Script

From Alexander Kiselev's Sept 2015 EICRoot examples:

```
void reconstruction()
{
  // Load basic libraries;
  gROOT->Macro("$VMCWORKDIR/gconfig/rootlogon.C");

  // Create generic analysis run manager; configure it for track reconstruction;
  EicRunAna *fRun = new EicRunAna();
  fRun->SetInputFile ("simulation.root");
  fRun->AddFriend    ("digitization.root");
  fRun->SetOutputFile("reconstruction.root");

  // Call "ideal" hit-to-track associator routine;
  EicIdealTrackingCode* idealTracker = new EicIdealTrackingCode();
  idealTracker->AddDetectorGroup("FWDGT");
  // Add a bit of fairness to the reconstruction procedure; smear "ideal"
  // momenta by 10% relative before giving hit collection over to KF fitter;
  idealTracker->SetRelativeMomentumSmearing(0.1);
  // Also smear a bit "ideal" vertex;
  idealTracker->SetVertexSmearing(0.01, 0.01, 0.01);
  fRun->AddTask(idealTracker);

  // Invoke and configure PandaRoot Kalman filter code wrapper;
  fRun->AddTask(new EicRecoKalmanTask(idealTracker));

  // This call here just performs track backward propagation to the beam line;
  fRun->AddTask(new PndPidCorrelator());

  // Initialize and run the reconstruction; exit at the end;
  fRun->Run();
} // reconstruction()
```

Equivalent art FHiCL configuration file

```
#include "fcl/minimalMessageService.fcl"

process_name : reconstruction

services : {
   message : @local::default_message
}

source : {
   module_type : FriendlyRootInput
   fileNames : [ "simulation.root" ]
   friendFileNames: [ "digitization.root" ]
}

outputs : {
   rootOut : {
      module_type : RootOutput
      fileName : "reconstruction.root"
   }
}

physics : {
   producers : {
      idealTracker : {
         module_type : IdealTrackingCode // Ideal hit-to-track association
         input : FWDGT                   // Consider only FWDGT clusters
         momentumSmearing : 0.1          // 10% momentum smearing
         vertexSmearing: [ 0.1, 0.1, 0.1 ]  // Vertex position smearing
      }
      recoKalman : {
         module_type : RecoKalman        // Kalman track fitter
         input : idealTracker            // using idealTracker clusters
      }
      pidCorrelator : {
         module_type : PidCorrelator
      }
   }
   reco_chain : [ idealTracker, recoKalman, pidCorrelator ]
   output_to_file : [ rootOut ]

   trigger_paths : [ reco_chain ]
   end_paths : [ output_to_file ]
}
```