Framework Taskforce Introduction

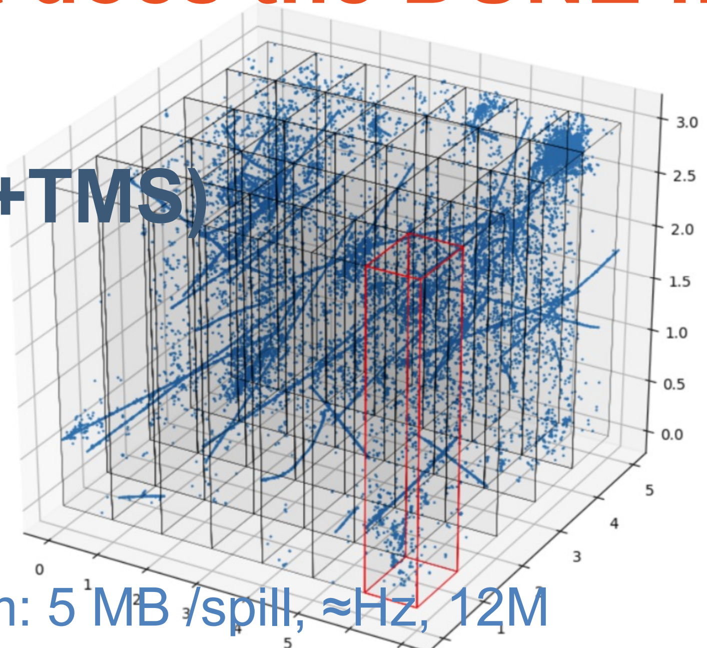
Callum Wilkinson
LBL

Mathew Muether
WSU

Framework Taskforce and Developer Workshop
June 27-28, 2024

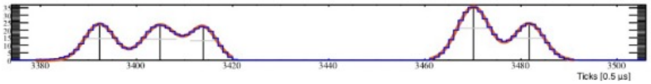
What does the DUNE framework need to do?

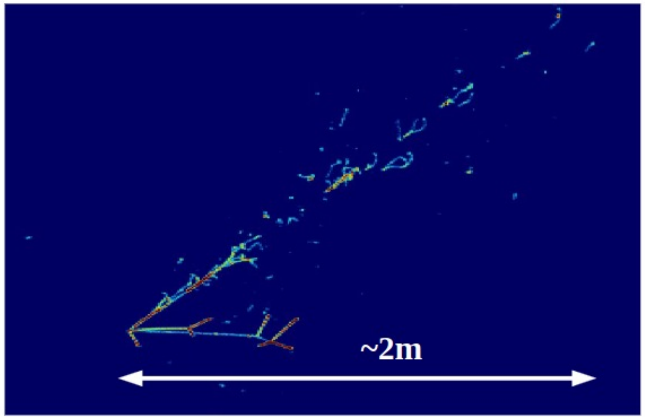
ND (LAr+TMS)



- Beam: 5 MB /spill, ≈Hz, 12M pixels, digitized hits, dense activity
- 15 M spills/year → significant computing requirements
- GPU-heavy custom sim+reco chain developed, current application in 2x2 prototype
- Joint reco across detectors.

FD



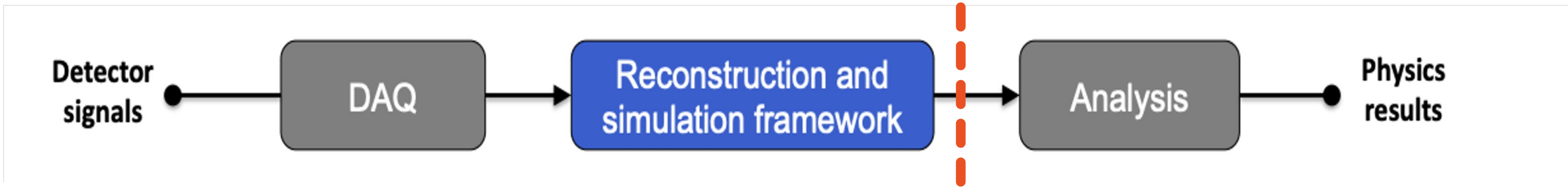


- Beam: ≈5 GB/trigger, ≈mHz, 3M wires, digitized waveforms, sparse activity
- Builds on sim+reco tools from current LArTPCs
- Unique challenge: very large (≈300 TB) supernova events,

What does the DUNE framework need to do?

- Deal with disparate data types with varying granularity:
    - ND/FD
    - Beam/HE/LE
    - Prototype data
- Support a wide variety of users
- Support a variety of existing sim+reco algorithms
- Utilize modern computing facilities *efficiently* – grid/HPC
- Be supported for the lifetime of DUNE
- **DUNE needs are different to existing experiments!**

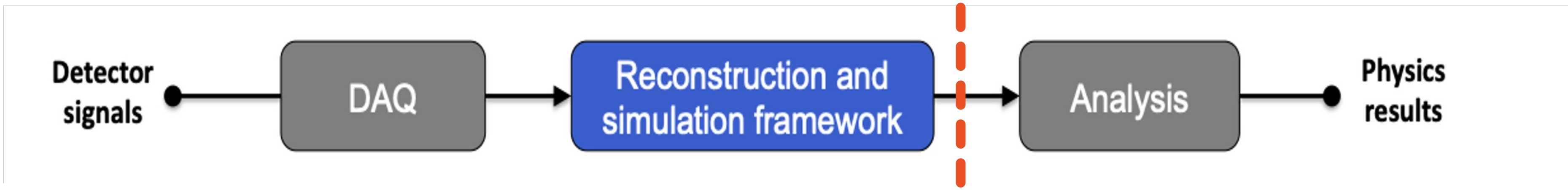
What doesn't the framework need to do?



- Deliberately noted that this is the sim+reco framework
- The framework will drive the offline processing loop up to the high-level analysis level → **CAFs**

  (Existing high-level analysis tools are able to consume these)
- However, it will not necessarily drive the high level analysis → decision/choice made by physics WG leads
- That said, we should be careful not to preclude high-level analysis extensions if possible

Aside: what are CAFs?



- **Common Analysis Format** files – lightweight analysis ntuples in ROOT format
- Defined interface between sim+reco and high-level analysis – both true now, and will be as we migrate to the new framework
- They are organized around a “StandardRecord” format, documented on a doxygen page
- Flexible structure, can be adapted to detector/reconstruction specific needs but common where possible, and centrally documented/managed

Framework timeline

Done!

- Initial FW requirements from taskforce
- Formation of framework development and management team
- Early development by design team
  - Iteration with FW taskforce
  - Workshop to finalize hand-off of requirements **– we are here!**
- Implementation specification + formation of framework stakeholders group **– end of Summer**
  - Independent expert review of design + requirements
  - Detailed milestones and resource loaded schedule at this point
- Code it up phase!
  - First prototype **~Summer FY25**
  - Phased implementation based on DUNE's physics needs

Iterative!

Framework Taskforce

- Taskforce set up in October 2023 to explore framework-related needs of DUNE
  - Chairs: Mathew Muether, Callum Wilkinson
  - +16 members across DUNE physics/detector groups
- Taskforce goals:
  - Review existing frameworks used by DUNE/HEP Experiments
  - Identify critical feature sets for new framework and produce a timeline with development, integration and deployment milestones
  - **Key deliverable:** Update 2018 requirements document
- Weekly meetings + Frameworks and Core Software workshop 11-13 Dec. 2023 at CERN
- **Requirements document v0 completed January 11**

Early iterations between devs and TF

- Iteration between this team and the TF since early February to digest and clarify requirements
- First step: develop a **design description** based on the requirements + rough time/resource/expertise estimates:
  - Map out architecture and design ideas
  - Design interfaces and workflows
  - Design how parts of the framework interact
  - Document design concerns, decisions and considered alternatives
  - This forms the “mental model” for users
- This has led to some clarifications of requirements, many additional use cases.
- Some important topics that require and deserve particular care

Aims for this workshop

- Finalize requirements relevant for core framework design
- Develop a comprehensive list of relevant DUNE use cases
- Develop a timeline for developing a rough design description
- Iterate on the broader development timeline as we have strong interest from review committees on this...

Framework “stakeholders” group

- A “stakeholders” team will be required to ensure adequate testing and feedback as framework is developed:
  - Workflow / data management
  - Databases
  - ND and FD sim/reco
  - Other physics WGs (users of the high-level CAF output)
  - Users of ND/FD prototype data
- **Test, review, iterate throughout development**
- This will be essential for ensuring DUNE needs are communicated efficiently to the developers
- And will also ensure that framework developments are rapidly deployed in DUNE workflows

Implementation phases

- The “code it up” phase will be split into work packages based on the implementation specification
- Those work packages will then be group based on needs to deploy the framework for different use cases
  - Integration of FD algorithms
  - Integration into workflow and production
  - Deployment for processing ongoing prototype data-taking
  - Deployment for reprocessing ProtoDUNE(-II) data
  - Integration of ND algorithms
  - Deployment for reprocessing ND-prototype data
- Prioritization into different implementation phases will be based on DUNE’s needs, and is worth keeping in mind here!

DUNE use cases/potential concerns

- Interaction matching across detectors (or major detector components) with varying timing, calorimetric, etc capabilities.
  - ND-LAr + TMS (and 2x2 + MINERvA) matching essential
  - Unclear if ND-LAr/TMS + SAND matching is necessary
- Multiple time series of files, where chunk size/period vary:
  - ND sub-detectors share a clock, but data rates differ
  - FD likely share a clock -- matching e.g. supernova triggers easy
  - Could potential phase II options require something different?
- Offbeam triggers -- different detectors might have a different strategy for triggering.
- Are there "stripes of data" throughout multiple files that a job would need to understand?

Elon Musk’s 3 Design Tips

1. **Make the requirements less dumb.**

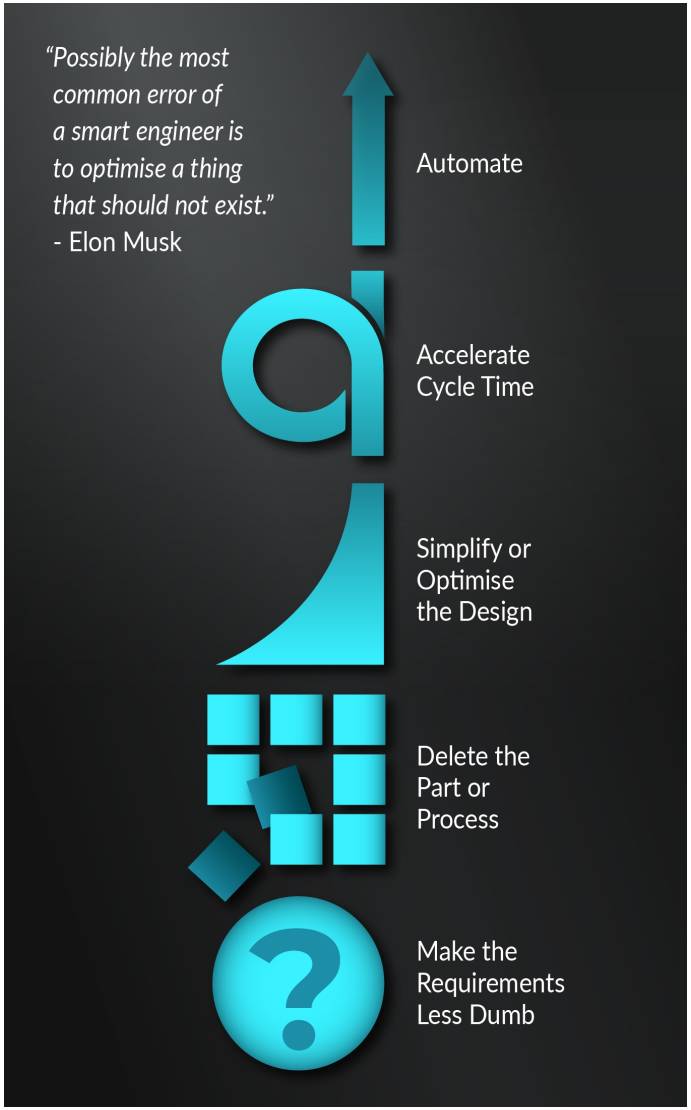
Question every requirement.

2. **Delete the part or process.**

Musk suggests starting lean and building up when and if required, but warns that the bias will be to add things "in case."

3. **Simplify or optimize the design.**

As mentioned earlier, Musk believes that the most common mistake that smart engineers make is to try optimizing something that should not exist.



“Possibly the most common error of a smart engineer is to optimise a thing that should not exist.”
- Elon Musk

Automate
Accelerate Cycle Time
Simplify or Optimise the Design
Delete the Part or Process
Make the Requirements Less Dumb

https://www.accountingtoday.com/opinion/elon-musks-5-step-process

Questions? Comments?

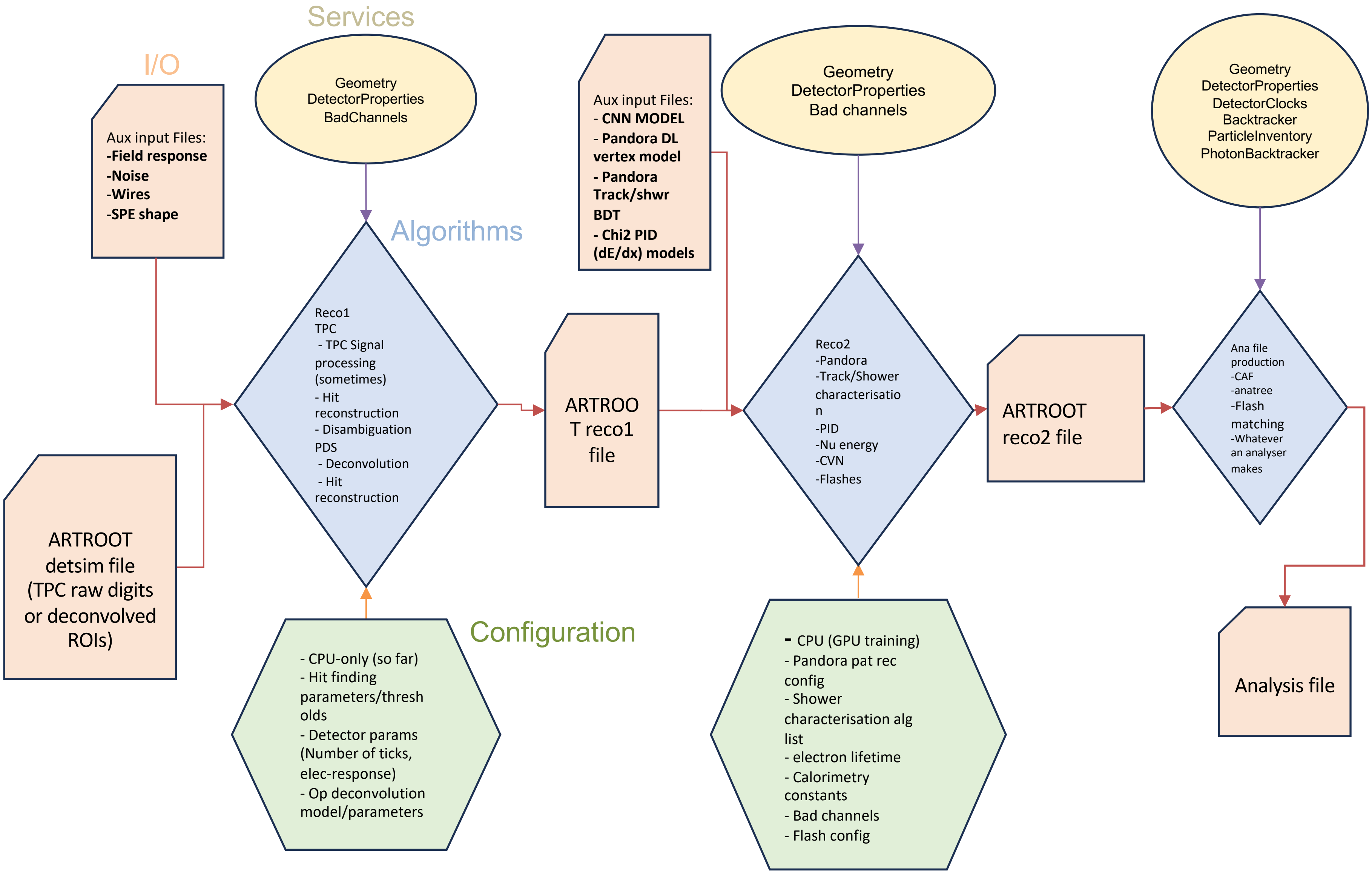
Development team and management

- Core computing framework is a DUNE US S&C deliverable:
  - Computing R&D L2: Peter van Gemmeren (ANL)
  - Framework L3: Kyle Knoepfel (FNAL)
  - + Persistency models L3: Barnali Chowdhury (ANL)
  - + Accelerator integration L3: Amit Bashyal (ANL), Meghna Bhattacharya (FNAL)
- Core frameworks development team from SciSoft at FNAL:
  - ~4 FTE – from the start of CY24
  - Led by Kyle (senior frameworks developer) under Adam Lyon, Division Director of DSSL (Data Science, Simulation and Learning)
- Funding for core development team and management is a high priority item under DUNE US FY25 ops funding request

Requirement document structure

- Breakdown of new document:
  - General requirements
  - Physics analysis requirements
  - Flexible Processing Unit (FPU) requirements
  - Reproducibility and provenance
  - Concurrency and multi-threading
  - Configuration
  - Services
  - Data I/O layer
  - (+ Glossary)
- General style is terse, with a hierarchy of requirements indicate with “MUST”, “SHOULD” and “MAY”
- Examples provided where deemed appropriate

FD sim + reco (ART supported)



I/O

Aux input Files:
-Field response
-Noise
-Wires
-SPE shape

ARTROOT detsim file (TPC raw digits or deconvolved ROIs)

Services

Geometry
DetectorProperties
BadChannels

Algorithms

Reco1
TPC
- TPC Signal processing (sometimes)
- Hit reconstruction
- Disambiguation

PDS
- Deconvolution
- Hit reconstruction

Configuration

- CPU-only (so far)
- Hit finding parameters/thresholds
- Detector params (Number of ticks, elec-response)
- Op deconvolution model/parameters

ARTROOT reco1 file

Aux input Files:
- **CNN MODEL**
- **Pandora DL vertex model**
- **Pandora Track/shwr BDT**
- **Chi2 PID (dE/dx) models**

Geometry
DetectorProperties
Bad channels

Reco2
-Pandora
-Track/Shower characterisation
-PID
-Nu energy
-CVN
-Flashes

- CPU (GPU training)
- Pandora pat rec config
- Shower characterisation alg list
- electron lifetime
- Calorimetry constants
- Bad channels
- Flash config

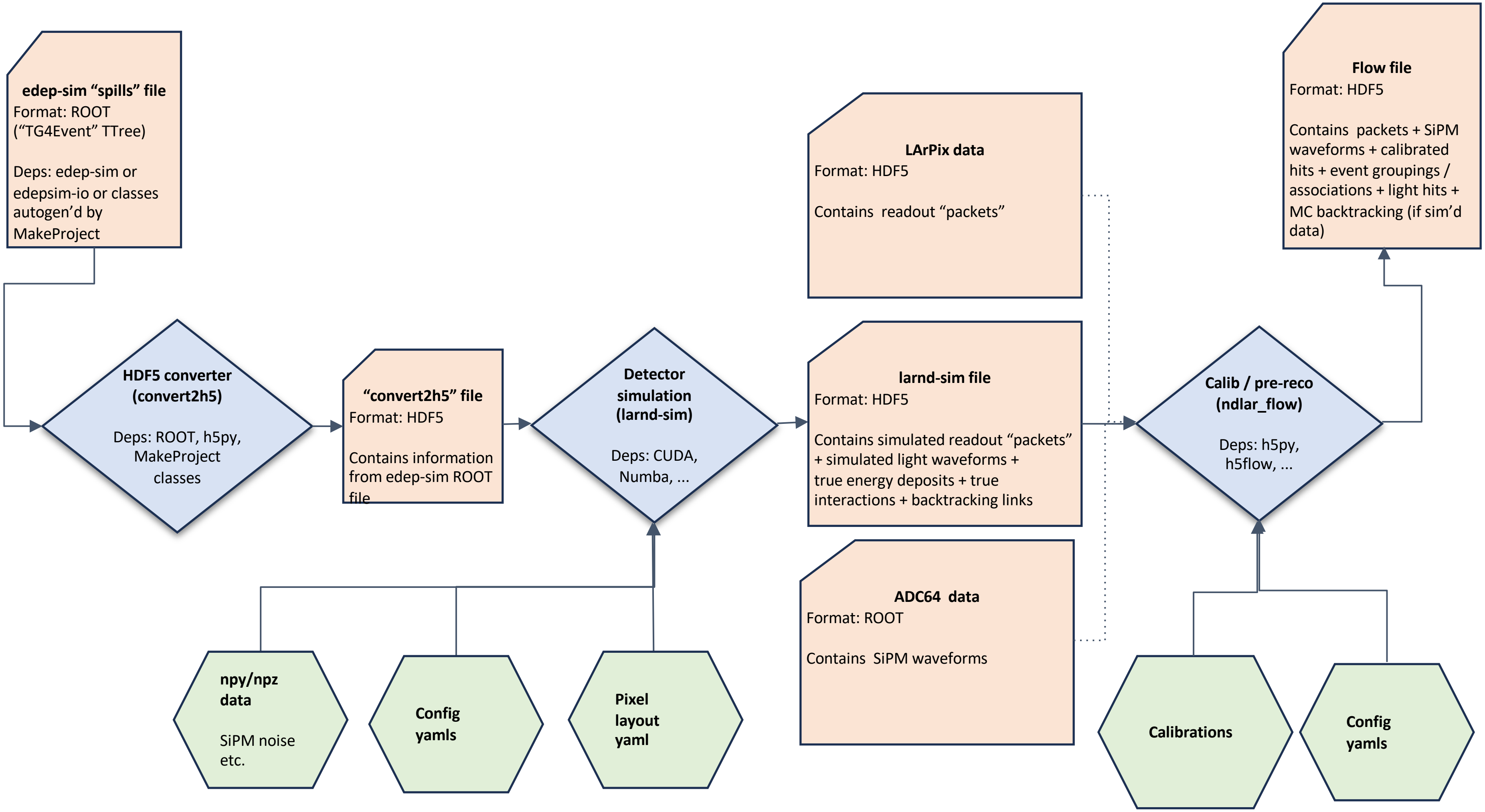
ARTROOT reco2 file

Geometry
DetectorProperties
DetectorClocks
Backtracker
ParticleInventory
PhotonBacktracker

Ana file production
-CAF
-anatree
-Flash matching
-Whatever an analyser makes

Analysis file

ND-LAr/2x2 simulation (No FW)



edep-sim "spills" file
Format: ROOT
("TG4Event" TTree)

Deps: edep-sim or edepsim-io or classes autogen'd by MakeProject

HDF5 converter (convert2h5)

Deps: ROOT, h5py, MakeProject classes

"convert2h5" file
Format: HDF5

Contains information from edep-sim ROOT file

Detector simulation (larnd-sim)

Deps: CUDA, Numba, ...

npy/npz data

SiPM noise etc.

Config yamls

Pixel layout yaml

LArPix data
Format: HDF5

Contains readout "packets"

larnd-sim file
Format: HDF5

Contains simulated readout "packets" + simulated light waveforms + true energy deposits + true interactions + backtracking links

ADC64 data
Format: ROOT

Contains SiPM waveforms

Calib / pre-reco (ndlar_flow)

Deps: h5py, h5flow, ...

Calibrations

Config yamls

Flow file
Format: HDF5

Contains packets + SiPM waveforms + calibrated hits + event groupings / associations + light hits + MC backtracking (if sim'd data)