cherenkov
telescope
array

Development Status

Design, what we have, and where we are going

Current Code Base

based on Scientific Python Stack (NumPy/SciPy/AstroPy/PyTables/Numba)

ctapipe

standard analysis tools
- ctapipe-process
- ctapipe-apply-model
- ctapipe-train
- ctapipe-fileinfo
- …

framework
- Data Input
- Event model
- Instrument Model
- HDF5 Data IO
- Configuration
- Logging
- Provenance

algorithms
- calibration
- image processing
- reconstruction
- atmosphere

py**IRF**

Cut Optimization
- Gamma-hadron
- Theta^2

IRFs
- Aeff
- PSF
- Edisp

Performance Metrics
- sensitivity
- angular resolution
- energy resolution

neo**bench**

benchmarking of sim processing

Performance checks

py**eventio**

low-level simulation data access

zfitsreader

low-level DL0/Event data input (from ACADA)

γπ A **Python** package for **gamma-ray** astronomy

IRF Format and output

Covers most but not all functionality…

Division of Responsibilities

⚠:
some structural re-organization could help (topic for later discussion)

DPPS-Common

- ctapipe.**io** (HDF5), ctapipe.**core**, ctapipe.**containers**, ctapipe.**instrument**, ctapipe.**utils**, ctapipe.**visualization**
- **zfitsreader** (with ACADA)

DPPS-DataPipe

- ctapipe.**reco**, ctapipe.**image**⚠, ctapipe.**atmosphere**
- parts of ctapipe.**calib** (with CalibPipe ⚠)
- **pyirf**
- **benchmark** tools
- DataPipe **Workflows** (with WMS)

DPPS-SimPipe

- ctapipe.io. **SimTelEventSource**⚠
- **pyeventio ?**

SUSS:

- **gammapy**

Code Standards

Standards:

- CTA Software Programming Standards Document
- Developer Guides
  - CTAO Developer Guidelines (work in progress, but useful for python already)
  - ctapipe developer documentation on readthedocs

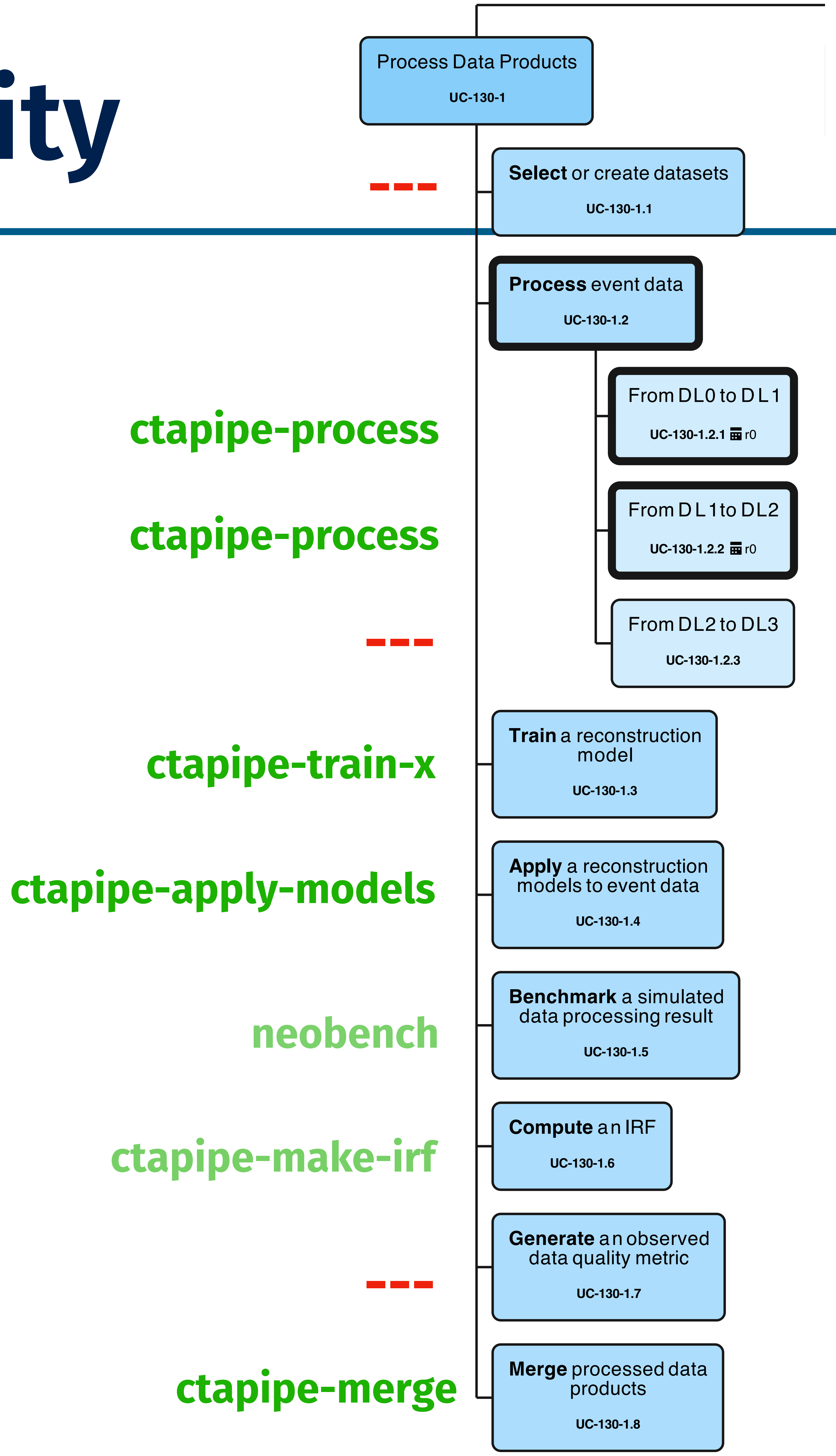
Practices:

- GitLab vs GitHub:
  - Github only for existing packages shared outside of DataPipe
- New Software Projects
  - Templates for python packages available in Computing GitLab space
- Quality Gates
  - CTAO SonarQube instance

Common core functionality

What we have now covers most of core functions

But not in full detail (missing things inside)



Process Data Products UC-130-1

--- Select or create datasets UC-130-1.1

Process event data UC-130-1.2

ctapipe-process — From DL0 to DL1 UC-130-1.2.1 ▦ r0

ctapipe-process — From DL1 to DL2 UC-130-1.2.2 ▦ r0

--- From DL2 to DL3 UC-130-1.2.3

ctapipe-train-x — Train a reconstruction model UC-130-1.3

ctapipe-apply-models — Apply a reconstruction models to event data UC-130-1.4

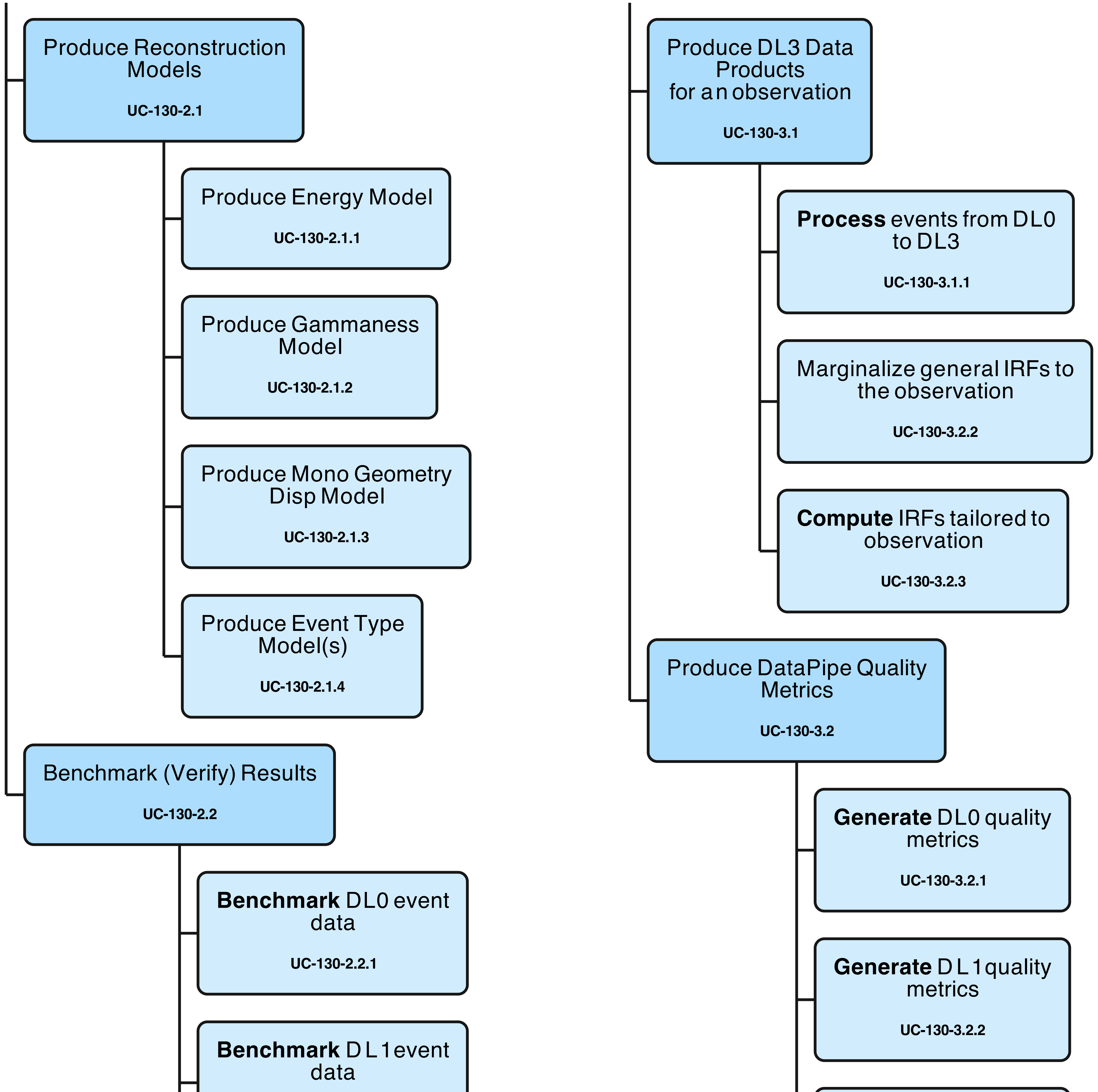
neobench — Benchmark a simulated data processing result UC-130-1.5

ctapipe-make-irf — Compute an IRF UC-130-1.6

--- Generate an observed data quality metric UC-130-1.7

ctapipe-merge — Merge processed data products UC-130-1.8

Higher-Level Functionality



Produce Reconstruction Models
UC-130-2.1

Produce Energy Model
UC-130-2.1.1

Produce Gammaness Model
UC-130-2.1.2

Produce Mono Geometry Disp Model
UC-130-2.1.3

Produce Event Type Model(s)
UC-130-2.1.4

Benchmark (Verify) Results
UC-130-2.2

Benchmark DL0 event data
UC-130-2.2.1

Benchmark DL1 event data

Produce DL3 Data Products for an observation
UC-130-3.1

Process events from DL0 to DL3
UC-130-3.1.1

Marginalize general IRFs to the observation
UC-130-3.2.2

Compute IRFs tailored to observation
UC-130-3.2.3

Produce DataPipe Quality Metrics
UC-130-3.2

Generate DL0 quality metrics
UC-130-3.2.1

Generate DL1 quality metrics
UC-130-3.2.2

ctapipe-train-energy-regressor

ctapipe-train-particle-classifier

ctapipe-train-disp-reconstructor

---

neobench

ctapipe-process

ctapipe.io.ZFITSEventSource

Data Models and Formats

R0 from Simulations:

- EventIO format
- Transformed into DL0 by SimTelEventSource

DL0 from Observation (ACADA):

- ZFITS format documented in R1/DL0 data models and ACADA-DPPS ICD

(R0, R1) DL1, DL2 used in pipeline:

- HDF5 format, Data Model specified in ctapipe.containers
- I/O implemented in:
  - HDF5EventSource/DataWriter (event-wise)
  - HDF5Merger/TableLoader (column-wise)

DL3 Event + IRF + Monitoring:

- FITS format (to be implemented)...

Processing Configurations

Category-B:

- Runs on-site data centers (unless data have already transferred)
- Starts at end of data taking, should be **finished by mid-next day**
- ≈20 observations per night

Category-C:

- Runs at off-site data centers
- Up to a **month to finish**
- can wait for sims, for example, or do fancier algorithms

Same software and workflows*, differ only by configuration for "quick and dirty" to "advanced final analysis"

*However, may skip some workflows steps if needed

Missing Pieces

Known Missing Functionality

Event Types

- Classify events into bins of *reconstruction* and *background* quality
- machine-learning step to predict the reconstruction uncertainties?

DL2→ DL3 conversion!

- Choose best reconstruction
- Categorize by Event Type
- Write FITS output in proper format

Simulating Calibration:

- R1 Calibration not fully simulated (no pedestal variances, etc)

DL1 Calibration for Observations (and simulations)

- R1 calibration happens in the EventSource (telescope responsibility)
- (re-)Calibration or refinement in the DL0→DL1 step doesn't exist

To be expanded during this workshop!

Observation processing:

- NSB-safe Image cleaning (using pedestal variances, times) → partially there, but see above
- Reading and Application of pointing corrections (translations and rotations)

Details of Cat-B/Cat-C

- alternate scenarios of top-level UC (not yet written)

Generation of SAG IRFs

- Uses potentially different reconstruction and calibration code…
- We need to develop a workflow for this

Potentially Missing Use Cases

To be expanded during this workshop!

Config Management:

- We will have many config files for different stages and types of processing
- How to manage and track them?
- Do we need a configuration UI?

Workflow Management

- Workload provides a lot of functionality for monitoring workflows.
- Is it enough?
- Do we need our own databases?

Useful Links

Suggest a change to function tree or use cases:

- open issue or merge-request on **dpps-use-cases** repo

Ask a question or Suggest a task for DataPipe group:

- Quick: use the #dpps-datapipe Slack channel
- Advanced/Tracked: