Common Software Framework for SIDIS Analysis of EIC Fast and Full Simulations

- Analysis repository and Overview
- Kinematics calculations
- Feature in Fast and Full Simulations Analysis
- Output Data Structures

Christopher Dilks
21 September 2021

Duke UNIVERSITY

ATHENA

General Purpose SIDIS Analysis Software

Github: https://github.com/c-dilks/largex-eic

- Dependencies: ROOT and Delphes
- Follow setup instructions in README.md, and tutorials in tutorial/
- Repository name “Largex-eic” is historical, could be updated
- Focus is on SIDIS, but some parts of the software can have common applicability in other working groups

---

Many tutorials are for fast simulations (for full simulations, change “AnalysisDelphes” to “AnalysisDD4hep”)

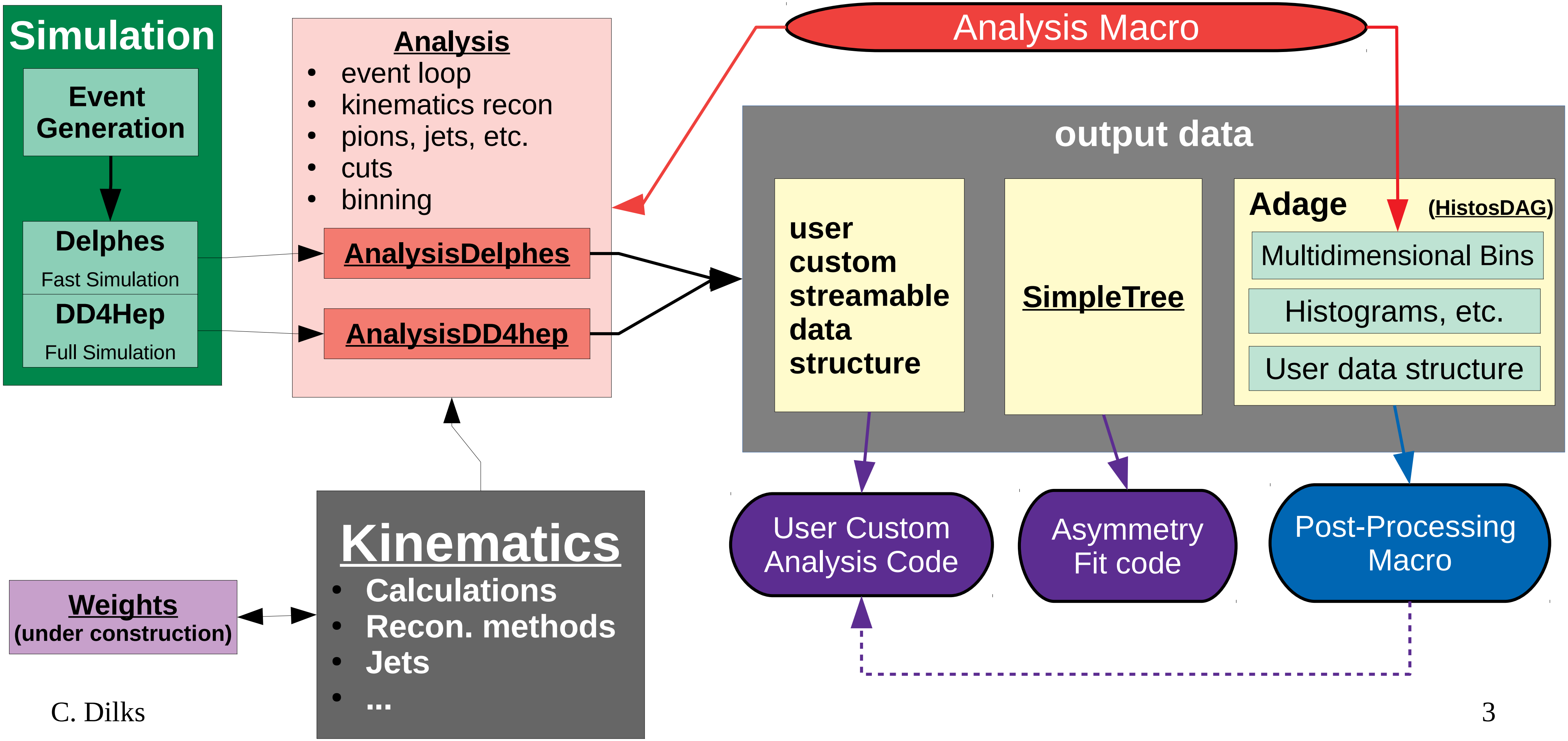
Example fast simulation ROOT file from Delphes, 5x41:

https://duke.box.com/s/0x83y9uz56vafvm9hxige7efov9z8taw

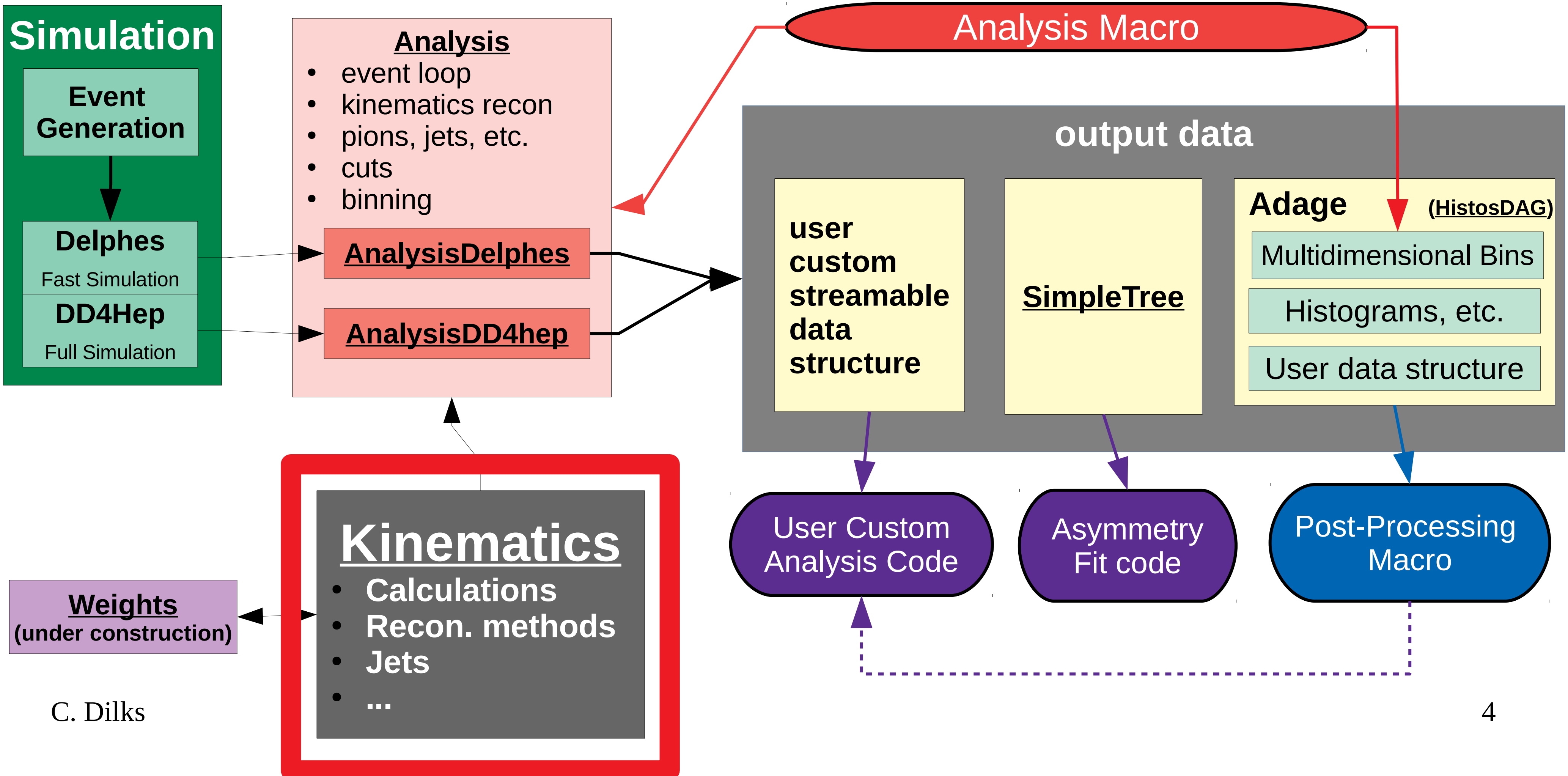
Download the ROOT file and store it in `largex-eic/datarec`

A hepmc file from Pythia is also provided, which you can run through Delphes (it is not the same data set as the example ROOT file)

Common Software



Common Software



Kinematics Class

- Contains kinematics reconstruction methods and calculations
- There are 2 instances: one for the reconstructed particle, and another for the true (generated) particle
- When reading each particle in the event loop, Kinematics calculations will be performed and variables will be set with the resulting values

Objects
- SIDIS kinematics {x, Q^2, y, p_T, q_T, … }
- Jet kinematics
- 4-momenta (in various frames)
- Spin

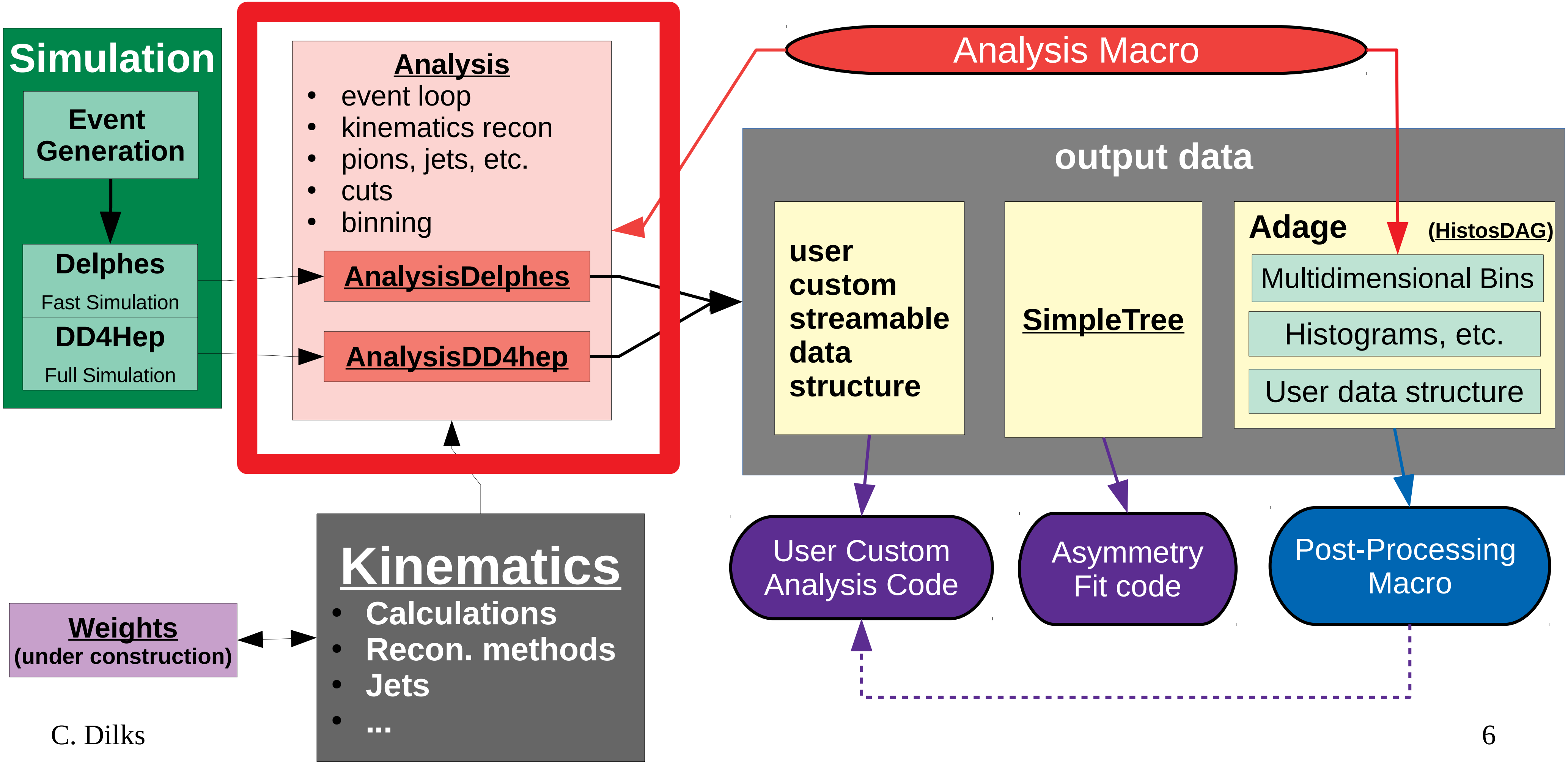
Methods
- Reconstruction of DIS variables (via electron, J.B., mixed, etc.)
- Reconstruction of Jet variables (fastjet)
- Boosts

Cuts
- Applied “globally”; could instead create a common “macro” to define these at the user-level
- Define your own “bins” for more cuts
- Common SIDIS cuts:

  DIS cuts:
  - $W > 3$ GeV
  - $y < 0.95$

  Hadron cuts:
  - $0.2 < z < 0.9$
  - $p_T^{LAB} > 0.1$ GeV
  - $x_F > 0$

Common Software



Summary of Implemented Features

AnalysisDelphes

- **Electron Finding**
  - Maximum momentum for reconstructed and generated
- **Hadronic Final State**
  - Tracks, energy flow
  - Used for some reconstruction methods
- **Reconstruct DIS Kinematics**
  - x,Q2,y,W,etc.
- **Jets Final State**
  - Energy flow, Fastjet, anti-k_T
- **Track Loop**
  - Calculate SIDIS hadron kinematics
  - Get weights
  - Stream to data structures
- **Jet Loop**
  - Calculate jet kinematics
  - Get weights
  - Stream to data structures

AnalysisDD4hep

- **Electron Finding**
  - Maximum momentum for generated
  - Calorimeter clusters, max momentum, given cuts:
    - Energy threshold
    - Isolation cone size and energy fraction
- **Hadronic Final State** (from `ReconstructedParticles` branch)
  - Tracks; used for some reconstruction methods
- **Reconstruct DIS Kinematics**
  - x,Q2,y,W,etc.
- **Track Loop**
  - Calculate SIDIS hadron kinematics
  - Get weights
  - Stream to data structures

Work in Progress; there is still work to do, features to add, validation, etc.

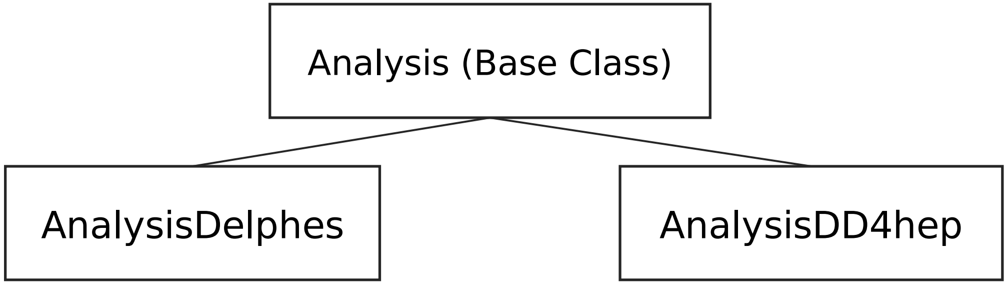
slide from Connor:

Current jet implementation in AnalysisDelphes

- Jet clustering using fast jet and Delphes energy flow objects with pT > 0.1 GeV
  - EFlowTracks, EFlowPhotons, EFlowNeutralHadrons
  - four-momenta from MC particle bank also clustered to get true jets
- Currently, semi-inclusive jets using anti-kT (R=0.8) algorithm clustered, then used to calculate other variables/fill histograms
  - $z_h, j_\perp, q_T$ etc.

slide from Sanghwa:

SIDIS Full Simulation Analysis



- Input files: RECO outputs

  - Truth information: mcparticles2
  - Scattered electron, neutral particles: Ecal and Hcal Clusters
  - Final state particles: ReconstructedParticles

- Shared the same output format as the Delphes analyzer

slide from Sanghwa:

Full Simulation Scattered electron identification

- Scattered electron identification done using calorimeter information with an isolation cut
  - Minimum energy for the cluster (default threshold: 10% of the beam energy)
  - Isolation cut (default: E_cone < E_e * 0.1 with the cone R = 1.0)

    - Loop over all the calorimeter clusters, summing up the energy within in a cone. Based on Miguel's python analysis: https://github.com/miguelignacio/calostudies
  - Cut values can be set from a macro

```
void SetEleEnergyThreshold(double e_threshold_) { fEThreshold = e_threshold_; }
void SetIsoConeRadius(double r_ ) { fIsoR = r_; }
void SetIsoCut(double isocut_ ) { fIsoCut = isocut_; }
```

* *Will be replaced by the common kinematic reconstruction output (once available)*

slide from Sanghwa:

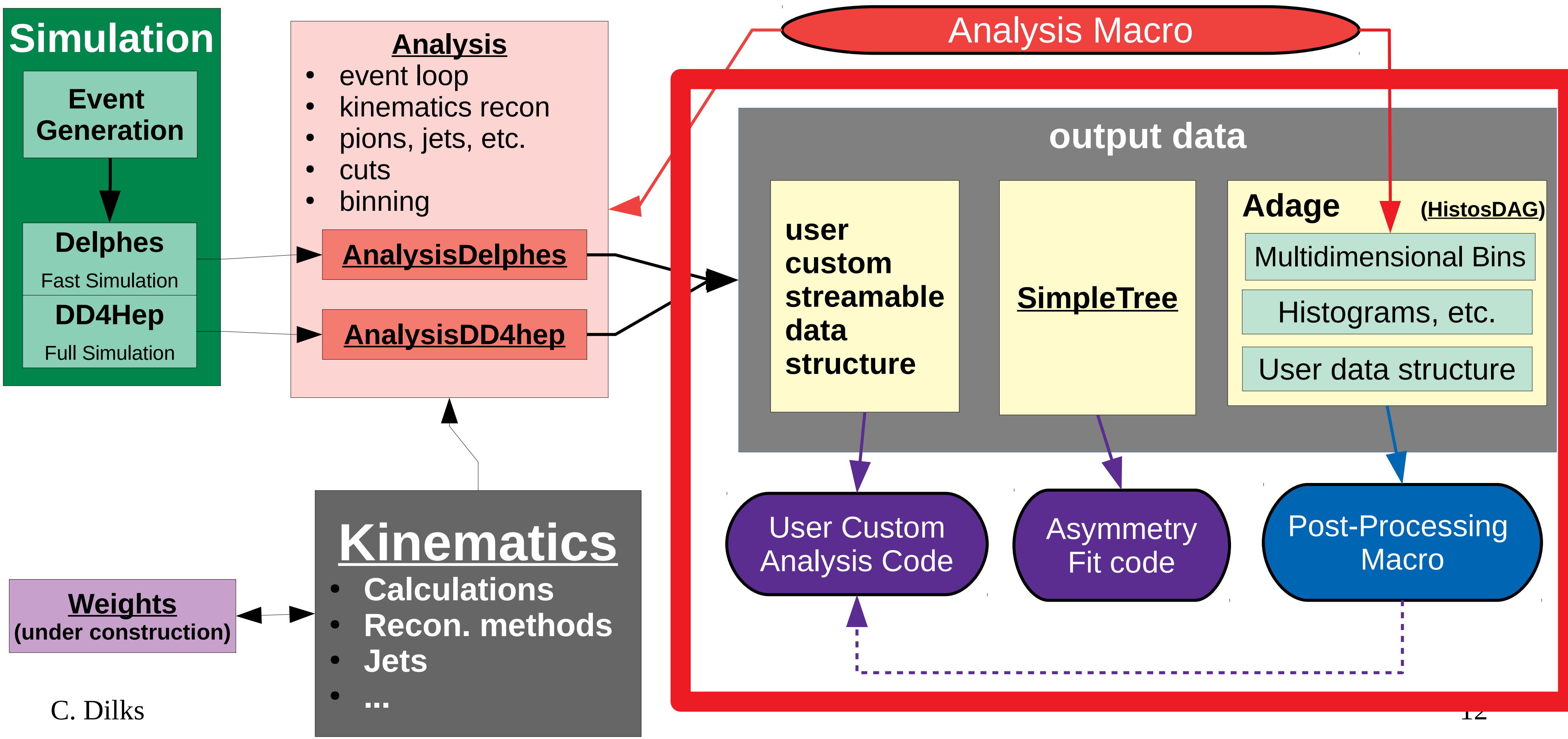
Full Simulation Analysis Improvement, to-dos:

https://github.com/c-dilks/largex-eic/issues/12

- [x] verify the output histograms
- [x] upload example running macros
- [ ] implement PID smearing
- [x] add other hadron truth information
- [x] fix hard-coded parameters (e.g. energy threshold for the scattered electron, better change to something like ~10% of the beam energy?)
- [ ] asymmetry check
- [ ] clean-up
- [ ] implement track-cluster matching (start with a simple projected distance check?)

→ This would be the one we could get some help from other WGs.
Any ongoing efforts or plan?

Common Software



Output Data Structures

- **SimpleTree**
  - Minimal, flat TTree for *specific* purpose of asymmetry fit code
  - Useful for other simple studies

- **Adage** – Analysis in a Directed Acyclic Graph Environment
  - Generalized multi-dimensional binning implemented in a graph data structure
  - Code (lambdas) can be attached to the graph and executed, streamlining differential studies
  - No restrictions on which variables or number of dimensions
  - Fine print: prototype implementation, use at your own risk, and validate the output!
  - Documentation: https://github.com/c-dilks/largex-eic/blob/main/doc/adage.md

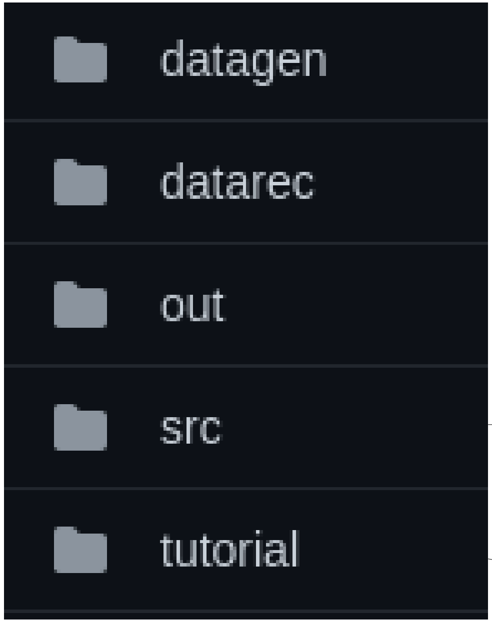
- **Custom User Data Structure** – Facilitate connections to your analysis code
  - Two options:

    (1) Stream to your custom data structure directly from the event loop in `Analysis`
    - Example: your own TTree or histograms
    - Example: stream back to existing simulation output data structures (trees)
    - Example: your own custom streamable

    (2) Add your data structure to Adage, for mult-dimensional binning studies

Code Repository

Github: https://github.com/c-dilks/largex-eic



generated data, reconstructed data, analysis and postprocessing output

Class definitions: see flowchart slide above for <u>class names (underlined)</u>

Tutorial Macros

Documentation:
– /README.md
– /tutorial/README.md

Contributions are Welcome

Git Workflow

- Fork or ask to be a contributor
- Pull requests:
  - Make <u>draft</u> pull requests for works in progress
  - Mark as <u>ready (open)</u> and someone will review (for compatibility / integration) and merge
- Issues
  - Report any bugs
  - Add any action item / task
  - Work on any open issue

Short Term Tasks

- Head-on frame boost
- Kinematics calculation validation / cross check
- Full Simulation
  - PID smearing
  - track-cluster matching

backup

General Procedure

underlined objects are classes (or macros)

- Choose your bins for each variable you are interested in; each bin of some variable x is specified by a CutDef, in a variety of ways:
  - Range: a<x<b
  - CenterDelta: |x-a|<b
  - Minimum: x>a
  - Maximum: x<a
  - No cut (full range of x)

- Bins of a particular variable x are collected into a BinSet (also called 'bin scheme'), where you can either:
  - Manually define each bin
    - Example: [Bin1: x<0.2] [Bin2: 0.2<x<0.5] [Bin3: x>0.5]
    - Example (note that overlapping bins are allowed!): [Bin1: full y] [Bin2: y>0.03] [Bin3: y>0.05]
  - Define an axis of bins: N bins between a and b
    - equal widths in linear scale
    - equal widths in logarithmic scale
    - any custom TAxis
    - Example: (x,Q2) bins with equal width in log scale

- **User specifies all Bins and BinSets in an analysis macro**

General Procedure

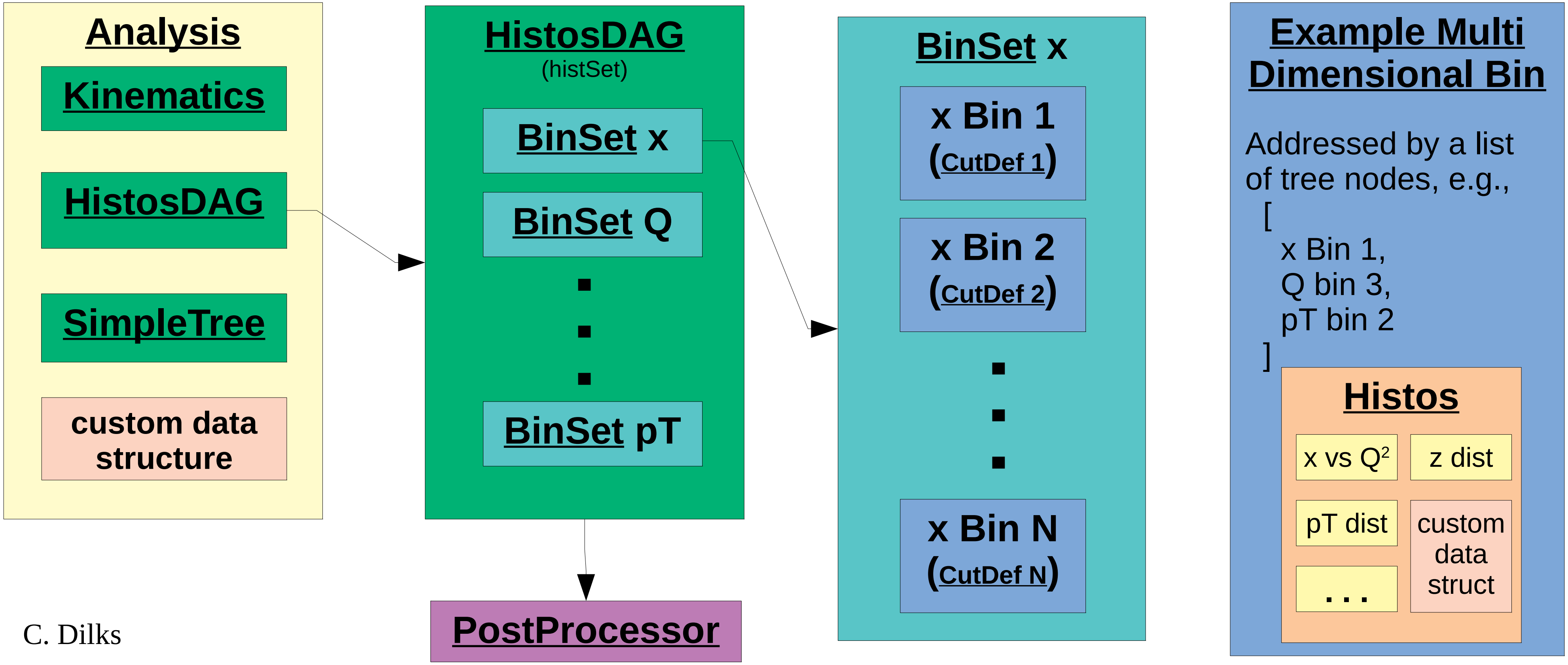
- Each multidimensional bin contains a Histos object
  - Set of user-defined histograms (1,2, or 3D)
  - Set of CutDefs associated with this bin
  - Settings for histograms (e.g., log scale drawing)
  - You are welcome to add your own data structures to the Histos class (or even inherit from it)

- No limit to number of BinSets, i.e. dimensions of your binning
  - You can only choose bins which are "available" in the Analysis class (see the constructor); you can also add your own (~3 lines of code)
  - Careful of the curse of dimensionality

- BinSet and Histos are streamable to ROOT files, which will happen automatically from an analysis macro
  - Analyze these with the PostProcessor class, which can do a variety of tasks:
    - Draw histograms in a specific format
    - Take ratios of histograms from two different bins
    - Dump averages of histograms for a set of bins and make a table
    - Add your own algorithms here
  - **PostProcessor is driven by a postprocessor macro, providing full bin-looping flexibility**

Adage: Analysis in a Directed Acyclic Graph Environment

- Tree (DAG) of multidimensional bins, where each bin contains a set of histograms, a "Histos" object



C. Dilks