Software Framework Planning

- Need specifications for (at least) each of
  - Simulation
  - Digitization
  - Databases
  - File formats
  - Reconstruction Framework
  - Calibrations
  - Physics Analysis
  - Computing Model
  - Level-3 trigger

Simulation & Digitization Wishlist

- Simulation
  - Package: GEMC (Hall B). Required SoLID modifications
    - ★ Add interface to SoLID database system
    - ★ Store relevant database parameters & metadata in output
    - ★ Ensure database consistency between simulation, digitization and reconstruction, esp. geometry
  - Should run within reconstruction framework. Generator(s) serve as event source.
  - Generators etc.: TBD by simulation group
- Digitization
  - Separate processing step
  - Should run within reconstruction framework
  - Develop long-term implementation after reconstruction framework in place
  - Include trigger emulation and hardware-level digitization (*e.g.* ADC reading instead of amplitude etc.)
  - Write CODA (EVIO) output (to test mapping & decoding)

Reconstruction Framework: Feature Wishlist I

- General: Based on ROOT, C++ throughout
- Try to combine best features of existing frameworks (Hall A/B/D, (s)PHENIX, EICroot, ...). Ideally, adopt one that is closest to our requirements.
- User experience
  - Scriptable user-interface (ROOT's interpreter) (?)
  - Configurable at runtime
    - ★ Dynamic configuration of analysis chain(s), expandable via plugins
    - ★ Flexible input sources and output formats
    - ★ User-configurable output contents
  - Data represented by data objects (streamable ROOT classes), produced by data producers (algorithms).
  - Support multi-stage analysis: DST files supported as both input and output
  - File formats: ROOT (all data classes), EVIO (CODA-type data only)
  - Self-describing output: DSTs contain database parameters and metadata from previous stages

Reconstruction Framework: Feature Wishlist II

- User experience (cont.)
  - Support condition testing modules (cuts, tests), dynamically configurable, within analysis chains. Skip processing if certain tests fail (or succeed).
  - Modules should be reusable without recompilation. Support inheritance from standard modules to extend/modify functionality.
  - Multiple instances of modules allowed, differing in configuration. Module instances should have names (unique?)
  - Analysis chain(s) should be very user-friendly to configure. All essential configuration information in one place.
  - Special requirements for parity experiments:
    Delayed helicity scheme, 1 event = 1 helicity group, etc.

Reconstruction Framework: Feature Wishlist III

- Technical
  - User-transparent multithreading
  - Probably should require ROOT 6, C++11
  - Minimize other software dependencies
  - Options to sync event stream at special events (it e.g. helicity flips, scaler events), or to preserve strict event ordering
  - Optimize for low memory per core (trend for new compute nodes), *i.e.* share read-only data (parameters etc.) across threads
- Simulation support
  - Propagate and access MC truth data for certain data classes (if input comes from MC)
  - Digitization data objects should contain references to truth data (hits, tracks, PID)
  - Option for substituting any input data with MC truth data (?)
  - Support for mixing data and MC events (?)

Some Concepts From JANA

- **Event sources**
  - Completely format-agnostic
  - Read events (whatever they are) from some sort of input (files, network, databases) into internal buffer (roughly a processing queue)
  - Multiple event sources may be defined
- **Data Objects**
  - Data structures representing information of interest (e.g. hits, clusters, tracks, PID likelihoods etc.)
- **Data Producers** ("Factories")
  - Algorithm classes
  - Produce their data objects exactly once per event (unless persistence requested, then once per run)
  - Request input data from other producers
  - Lowest level data ultimately retrieved from event sources
  - Run in threads, operating on thread context data