DIRC: Technical briefing

- **Background: Ring Dictionary**
- **Classes `PmcDircModel` and `PmcDircResponse`**
- **Interaction with tracking code**
- **Caching of results**
- **Code organisation**
- **Open issues**

Ring Dictionary

- Lookup table for average number of photons based on hit location, momentum, particle type, and angle.
- Populated by a GEANT4 simulation of the *BABAR* DIRC.

Cherenkov Angle and Error Calculation

- Uses angle calculation based on β and *BABAR* DIRC material.
- Error calculated in two parts: (a) achromaticity error based on *BABAR* parameterization, and (b) momentum variance error based on *BABAR* parameterization. Returned value comes from combination. Would probably like to include only achromaticity and geometry related errors at this stage; add reco-related errors at a later stage.

`PmcDircResponse`

- **Very trivial struct! Three public variables:**
- **Number of photons:** `numPhotons`
- **Cherenkov angle:** `cherenkovAngle`
- **Error on Cherenkov angle:** `cherenkovError`

`PmcDircModel`

- **Current model is wrapper for Ring Dictionary code.**
- **Easy to add new models: Register with a keyword and factory method, then lookup anywhere in the code by keyword. Allows change of DIRC model in tcl script.**
- **Overloaded method `getDircResponse` has three interfaces for getting `PmcDircResponse`:**
  - `const HepPoint& location, const Hep3Vector& momentum`
    `double momVariance, int charge,`
    `PdtPid::PidType partType, double endRadius`
  - `PacSimTrack const * const trk`
  - `const PacSimHit& hit`
- **Expect client code mainly to use `PacSimTrack` or `PacSimHit` methods.**

Tracking interaction

- **`PacDircMeasurement` class inherits from `PacMeasurement`**
- **DIRC detector elements are assigned a `PacDircMeasurement` object on construction - modified the `PacCylDetector` constructor accordingly.**
- **When a `PacSimHit` is created, the `PacDircMeasurement`'s method `createRing` is called, returning a `PmcDircResponse` for caching.**

Result caching

- Not yet clear what final form of storage will be.
- Currently, convert `PmcDircResponse` to `DrcPidQual` and assign to `BtaCandidate` created by tracking code.
- Advantage: Code exists and is understood.
- Disadvantage: Not very lightweight.
- Not clear if `BtaCandidate` and friends will be used.
- Likely design: `PmcDircResponse` object will be owned by event, and a map will associate it with appropriate `PacSimTrack`.

Code organisation

- Plan is to create a new package, `PacDirc`, for DIRC-related code:
  - `PmcDircModel` (to be renamed `PacDircModel`) and future subclasses.
  - `PacDircMeasurement`.
- Detector-building code resides in `PacTrk`.
- Possible future change: Let the central detector-building code be a register of callback hooks, and thus allow clients to do the detector building without changing `PacTrk` code? May be too general for our purposes.

TODO list

- For the end of June
  - Understand and implement final form of result caching.
  - Move all code into `PacDirc` package.
  - Provide full documentation
- Longer term
  - factorize error into DIRC contributions (to be part of `PmcDircResponse`) and reconstruction contributions (to be implemented elsewhere).
  - account for DIRC bar geometry (cracks) in greater detail.
  - account for track interactions inside the DIRC.