BaBar's Beta Analysis Toolkit

Need analysis happening early to

- ¥Get reco products tested
- ¥Get people comfortable with language, environment, etc
- ¥Exert common force on analysis tecnhicalities
- ¥Start building collaboration in physics

Need to simplify and regularize physicistÕs view of the system

- ¥Must ease transition to C++, OO
- ¥Resource limits => must be self teaching
- ¥ Glass box reconstruction required for acceptance

ÒBetaÓ is the BaBar package to do this

Work of a large number of people

Charles Chiang, Pete Elmer, Paul Harrison, Rahul Kapur, John LoSecco,
Andrei Salnikov, Gabriella Sciolla, Gloria Vuagnin, John Walsh, Charlie Young

Simplicity is the key metaphor

¥Familar concepts save documentation:

Candidates , Operators , etc

¥Supports procedural code for analysis

¥Complexities invisible to the average user

Available to experts who are extending Beta

¥Making things easy, simple appearing is very hard

¥Deliberately _not_ enforcing a single approach to analysis

at significant cost in power

hope is that these will naturally evolve on this base

"A tool that isn't sharp enough to cut you badly isn't sharp enough to use."

--A basic law of computer programming

"However, the sharp edges should NOT be on the handle."

--Krasno's corollary

Beta

The Beta package is an analysis tool kit for use with Babar data.

- ¥It owes a lot to it's predecessors, most notably ALPHA from ALEPH.

Its primary goal is to provide a simple, common basis for writing detailed physics analysis programs.

- ¥It must be very easy to use, with easy to understand metaphors for common tasks, and without requiring the user to master the archania of the C++ programming language.
- ¥It must be complete, in the sense that it permits the analyst to examine anything available within the reconstruction processing and output.
- ¥It must be common, which means that it must be usable on all supported machines, with all our sources of data, at all levels of detail.
- ¥So long as the required information is available, an analysis routine should run on Aslund, bbsim and real data output, and regardless of how sparsified and compressed.

Regretably, only if time and resources are available can we address:

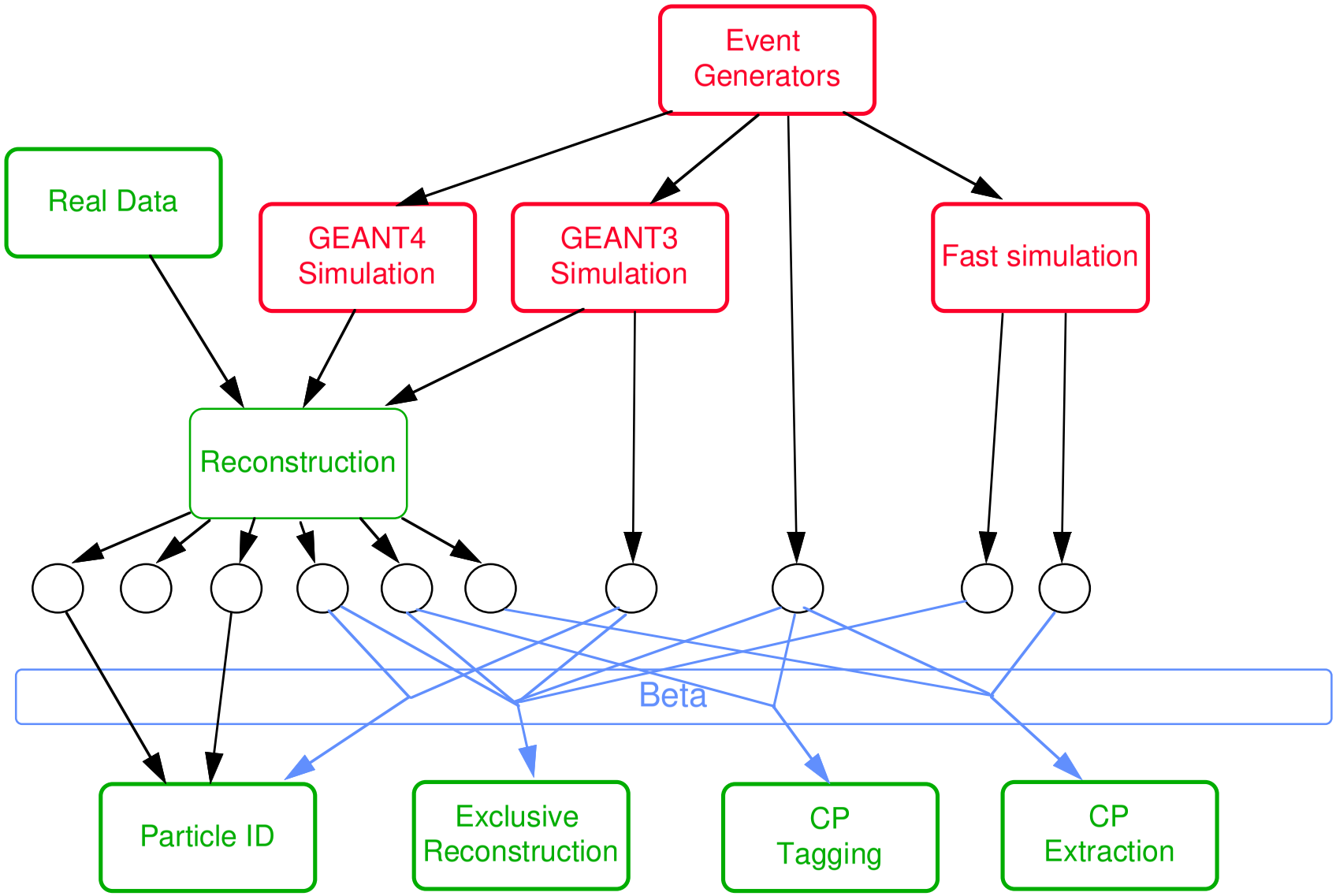
- ¥It should form a portable base for higher level analysis toolkits
- ¥It should be useful to an audience outside Babar

Basic ideas

Candidate: A candidate is the Beta representation of what is known or assumed about some particle that may have existed in the event in question. Charged tracks can be candidate pions, neutral clusters can be candidate photons. The output of vertexing two candidates is another candidate, representing the "particle" that might have decayed to those two tracks. There are many types of candidates, from different sources and with different implementations. But all of them provide the same, complete interface, so they can be used interchangeably.

Operator: An operator can combine one or more candidates to form new ones. Vertexing, mass constrained fitting, and simple 4vector addition are all operators. The analyst selects the operator needed at particular stages in the analysis, trading off accuracy and completeness for speed and flexibility.

Finders: A basic operation of analysis is creating acceptable candidates for a particular reaction from from scratch. For example, K0s candidates can be made by vertexing pairs of pion candidates, vertexing, and applying cuts. To the extent that these are reusable in other analyses, they become "finders". We expect to have a whole spectrum, ranging from general V0 finders, to K0s, D0 and pi0 finders, down to ones that are very specific to a particular analysis.


Event Generators
Real Data
GEANT4 Simulation
GEANT3 Simulation
Fast simulation
Reconstruction
Beta
Particle ID
Exclusive Reconstruction
CP Tagging
CP Extraction

Presently available I

(From http://www.slac.stanford.edu/BFDOC/www/Beta.html)

Access to charge and momentum

¥Quoted at default position, eventually IP

¥Access to underlying info

```
double           charge() const
HepLorentzVector p4()     const
Hep3VectorD      p3()     const
double           p()      const
double           mass()   const
double           energy() const
```

p(Point) support in BtaCandidate

¥Not meant to be as fast, accuracy varies by type

¥Subclass implementation for charged tracks (closest point)

¥Neutral implementation still needed

Presently available II

Genealogy and combinatorics support

¥Access to mother/daughter relations

```
nDaughters(), daughterIterator()
nMothers(), motherIterator()
```

¥Possible to have > 1 mother

¥Mother/daughter represents included within the hypothesis , can test

```
overlaps()
```

Access to optional information

```
const TrkRecoTrk*      recoTrk()  const
const AbsRecoNeut*     recoNeut() const
const BtaAbsVertex*    decayVtx() const
const PdtEntry*        pdtEntry() const
const BtaMcTruth*      mcTruth() const
const AbsPidSummary*   pidSummary() const
```

Presently available III

Particle type info

¥Can ask, set Particle Data Table entry

```
const PdtEntry*        pdtEntry() const
void setType(PdtEntry* pdt);
```

¥Need to tie into setting mass?

¥Need to develop conventions for use?

¥Need access to particle lists by type?

Accessor as another type of finder

Presently Available: Operators

Add4 and Vtx operators

- ¥Combine candidates as 4vectors or via spatial vertexing
- ¥Sequential vertexing present, but seems to be giving wrong answers!
  - K0 mass resolution not as good as it should be

Operator +, operator +=

- ¥Adds 4 vector at IP, charge, to make a new candidate
- ¥Doesn t provide genealogy, other complicated results
- ¥Intended for fast, rough cuts
- ¥Hard to do better due to expression handling, argument types
- ¥Probably a mistake
- ¥**WILL BE CHANGED TO RETURN A 4-VECTOR OBJECT**

Presently Available: Vertex Class

Represents the vertex at the decay end of a particle

```
virtual HepLorentzVector v4()
virtual HepPoint point() const
virtual double chiSquared()
virtual nDof()
```

¥No all candidates have vertex information, and it can arrive later

¥Implemented by specific classes for output of vertexing operator, etc

Partially Present: Candidate Features

Errors

¥Designed / defined for position, momentum and vertex

¥Needs code

¥Also needed on vertex

p(Position) information

¥Not done right for calorimeter neutrals

Aslund case needs a dedicated subclass or change way loaded

¥Vertexing output needs to be checked

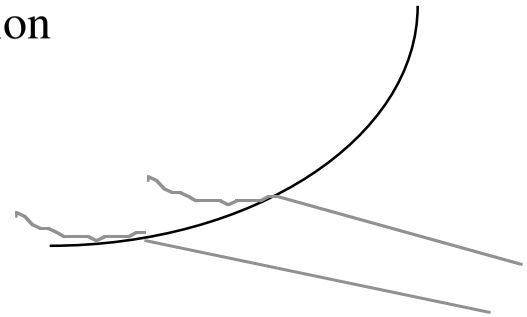
Reconstructed neutrals

¥Need IFR, EMC object definitions

¥Need code to convert to 4vector as function of position

¥Need connection to vertexing

Note: some cases make no sense!



Two particle mass bump

```
// quick plot of di-neutral spectrum for reference
HepConstAListIterator<BtaCandidate>
      iterGamma(*(anEvent->getBtaRecoNeutralAList()));
BetaOpAdd4 op;
const BtaCandidate* gamma;
const BtaCandidate* gamma2;

while ( gamma = iterGamma() )  {
    HepConstAListIterator<BtaCandidate
                          iterGamma2(iterGamma);
    while ( gamma2 = iterGamma2() )
       _neut->accumulate(op.combine(*gamma,*gamma2).energy());
```

Used to have operator +, but removed it:

¥Can t get arithmetical properties: A+(B+C) › (A+B+C)

(plot of energy of pairs of neutrals)

Vertexing K0 -> π+ π-

```
BtaOpVtx c;
while ( trk1ptr = iter1() ) {
  // start a further loop over the tracks
  HepConstAListIterator<BtaCandidate> iter2(iter1);
  while ( trk2ptr = iter2() ) {
    // check charge combination
    if ( (trk1ptr->charge())*(trk2ptr->charge()) == -1 ) {
       newTrkptr = c.create(*trk1ptr, *trk2ptr);
       // use the result only if a valid vertex was found
       if (0!= newTrkptr && newTrkptr->decayVtx()) {
            // add candidate to list
            _ownList.append(newTrkptr);
            pHist->accumulate( newTrk.mass() );
       } else {
            // not keeping, so must delete
           delete newTrkptr;
       }
```

Vertexing

HepTuple wrappers

Very procedural bias!

(plot of K0s reconstructed decay distance)

(Rose diagrams of the class structure)

¥Bta Candidate, BtaOperator are center of web of responsibilities

¥http://www.slac.stanford.edu/BFROOT/doc/Computing/Reconstruction/CHEP97/Beta/class.eps

(Rose diagram of the 7 class categories)

¥http://www.slac.stanford.edu/BFROOT/doc/Computing/Reconstruction/CHEP97/Beta/category.ps

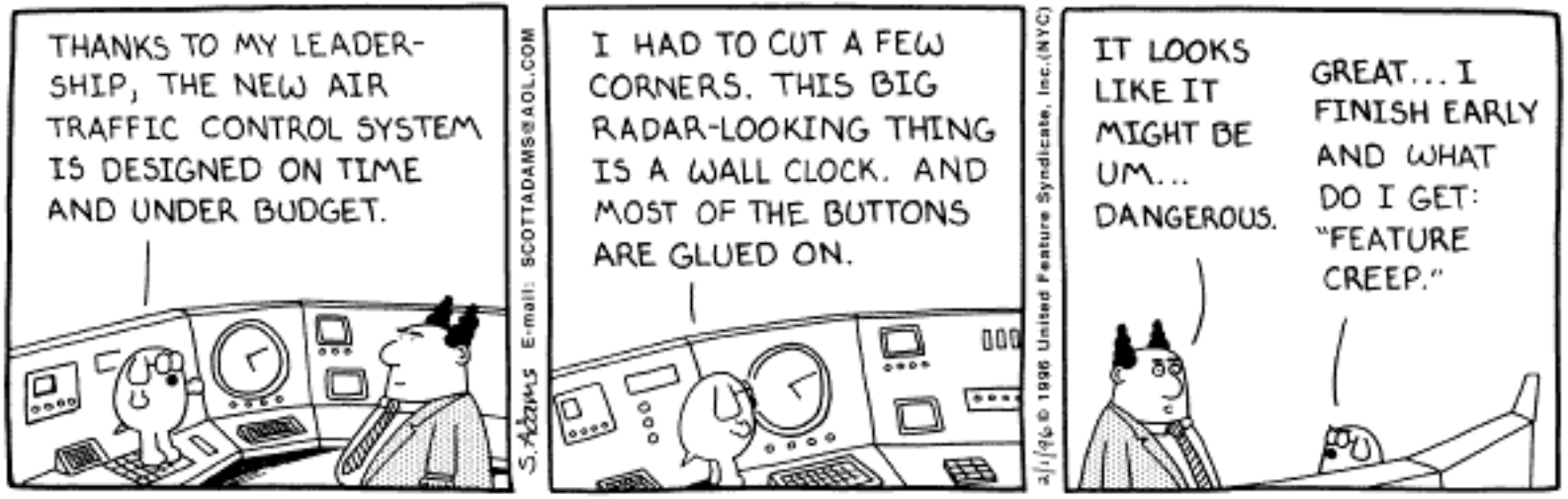
But it doesn't do ....

- Fast Ntuple analysis
- Your favorite fit
- Higher-level languages for analysis

"Here's how to start adding it ...."

For those outside intended scope: "Build those on top of Beta"

- More stable interface
- Access to all data sources



Copyright © 1996 United Feature Syndicate, Inc.
Redistribution in whole or in part prohibited

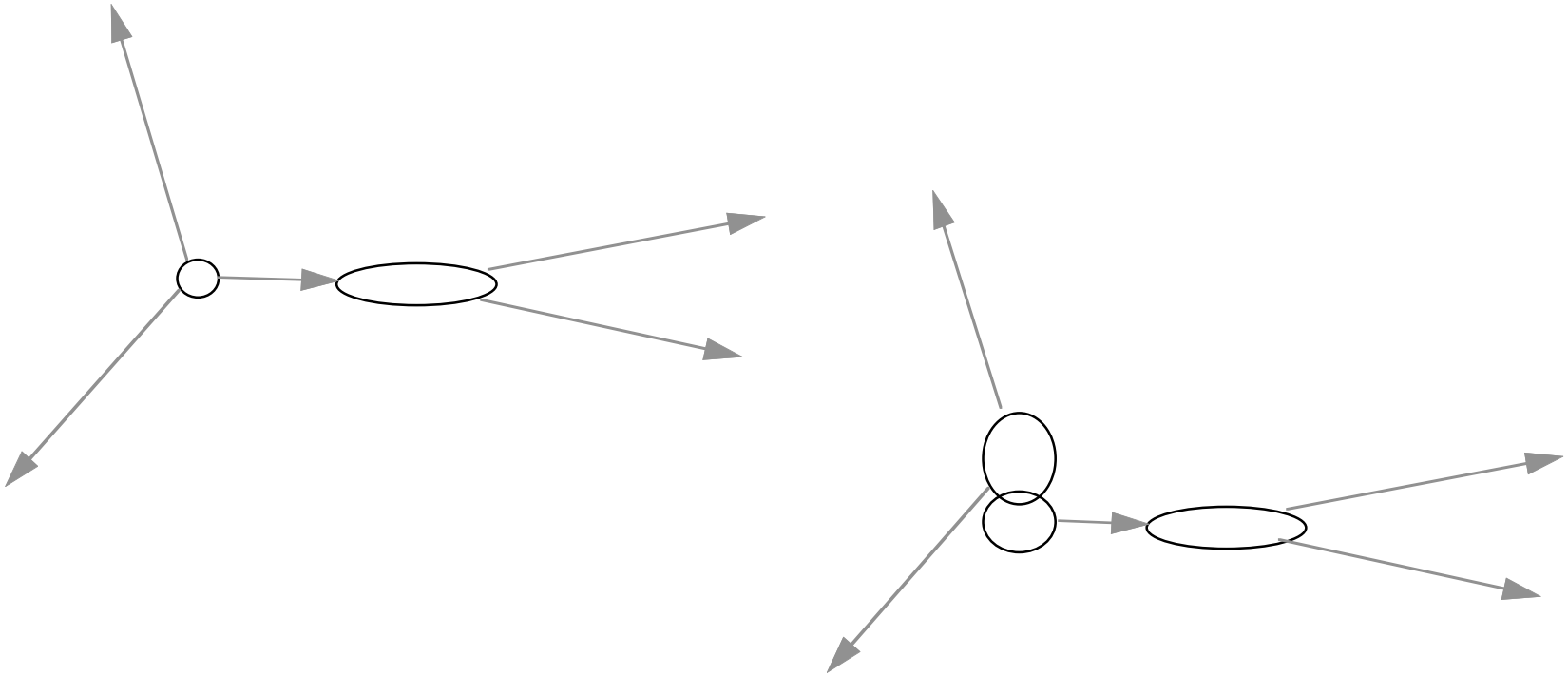
Does let you do interesting things:

K0 -> π+ π+

¥Background study?

¥Or stupid mistake?

B-> J/psi K0s as vertex(vertex(mu+,mu-), vertex(pi+, pi-)



Status & Prospects

Currently in use

- ¥About 35 users have looked at it
- ¥About 15 have asked questions
- ¥>= 12 have shown plots
- ¥Used for some QA functions
- ¥Physics coordinator has been supportive

"I have never seen anything fill up a vacuum so fast and still suck."
Rob Pike, on X11

Evolving solution to changing needs

- ¥ Ready to Use vs Cooperative Growth
- ¥Extending into other sources, uses
- ¥Examples, user docs slowly arriving
- ¥Features come and go

"Use the source, Luke"
O.W. Kannobe

Creating a self-supporting user community

Physics community has to be involved

- ¥Need to make people effective
- ¥Need to create momentum

"If you build it, they will come"
Voice in Kevin Costner's head

Related Talks at CHEP97

Architecture of the BaBar Reconstruction System: Practical Issues

(Tuesday 10:30 P5)

Panel Discussion: Transition to OO?

(Wednesday AM)

The BaBar distributed development / centralized release system

(Wednesday 18:00 P3)

Applying Object-Oriented Software Engineering at the BaBar Collaboration

(Thursday 14:00 P3)

An object-oriented tracking system for BaBar

(Thursday 14:30 P5)

An object-oriented extended Kalman filter tracking algorithm

(Thursday 16:00 P5)

Beta: A high level toolkit for physics analysis

(Thursday 17:00 P5)