

BTeV-doc-1647

The BTeV Software Tutorial Suite

Rob Kutschke, Fermilab

Presented at

CHEP03, San Diego

March 27, 2003

What is BTeV?

- At the Tevatron p-pbar collider, at Fermilab:
 - 1 arm forward spectrometer.
 - Beauty and charm physics:
 - Precision measurements.
 - Exhaustive search for new physics.
- BTeV is a part of broad program to address fundamental questions in flavor physics.
- <http://www-btev.fnal.gov>
- Analysis patterns will resemble B-factory, fixed target charm, and b-physics at CDF/D0.

A Brief History of BTeV

- June 2000: Stage I approval from Fermilab.
 - Fall 2001: funding situation deteriorated.
- May 2002: Stage I approval reaffirmed.
 - Descoped detector (one arm).
 - Offline computing to be supplied via universities.
- October 2002: Passed internal FNAL review
 - Addressed cost/schedule/technical risk.
- March 2003: P5 review this week.
- Fall/Winter 2003? Lehman baseline review.
- 2008? Start of physics running.
 - Using Fermilab's projected funding profile.

Some Unique Aspects of BTeV

- Track and vertex reconstruction in L1 trigger.
- L2/L3 Trigger farm, $O(2500)$ processors reused for offline reconstruction/simulation/analysis.
 - $\approx 2/3$ of cycles will be available.
- No traditional central computing center:
 - Cycles/disk from the L2/L3 trigger farm.
 - Resources supplied by universities.
 - Tapeless!
- Highly distributed computing must be ingrained.

Current State of Software

- Codes used during design and approval stage have accomplished their mission and will be phased out:
 - Simulations based on Geant3 and MCFast.
 - Many sophisticated reconstruction algorithms:
 - Kalman filter; ECal reco code derived from X-Ball and CLEO.
 - Mixture of C/Fortran/C++
 - Weaknesses:
 - Many, many assumptions and idealizations hard coded.
 - Requires fluency with many magic words and phrases.
 - Documented by oral tradition at coffee time.
 - Data and meta-data handling will not scale to large data samples.

Moving To the Future

- Design and build modern infrastructure in which the next generation physics software will live.
- **Learn from experiences of previous experiments.**
- Expect the first new physics codes in a year or so.
- By then we must:
 - Understand the major components and how they interact with each other.
 - Implement enough of the core infrastructure to allow (quasi) independent development of the many pieces.

For Whom are we Designing?

- Experienced physicists
 - Many without C++ background
 - Must not be marginalized by long learning curves.
 - Many with minimal C++ training.
 - Have many bad habits (MOAO, lots of static casts).
 - A few who only need a good reference manual for BTeV specific software.
- New physicists:
 - Broad spectrum of programming backgrounds.

Outline of the Software Model

- Framework/Module/Service + Event Data Model(EDM)
 - Physics code lives in modules, which are called by the framework in the requested order.
 - Events passed from module to module by framework.
 - Modules may respond to any of:
 - Start/End of (Job, Run, Run Segment).
 - Event – many sub elements.
 - And others to be defined.
 - Services provide information to modules or provide monitoring info to log files:
 - Eg. Message logger, Geometry Manager, Calibration Manager, timers, memory use monitors, ...

What We Have Now

- First drafts of:
 - Framework, module, run time configuration and EDM.
 - Borrowing many ideas, but little code, from D0 (CDF).
 - Provenance included in EDM.
 - Track, shower, vertex classes which live in new EDM.
 - IO module which reads legacy events and reformats them into a new style event.
 - **Tutorials to illustrate the above.**
 - Not doing modern persistency yet.
- Concentrated so far on designing the interfaces which will be seen by physicists.

Mandate for the Tutorials

- Well documented Tutorials which:
 - Allow physicists with little/no C++ background to do real work as quickly as possible.
 - Sell the software (C++) to the collaboration.
 - Always “just work”.
 - Form an index to the existing documentation.
 - Introduce the supported tools, eg. fitters.
 - Teach good C++ practice.
 - Rigorous type safety and const correctness.
 - Will be part of the nightly validation suite.

What's in a Tutorial

- Actual working code which does something in the day to day working life of a physicist.
 - Eg. Occupany maps ... complete simulated analyses.
 - Makes histogram/table or other output which is familiar to physicists and which can be compared to a reference.
 - The reference histogram/table is included in the distribution.
- Minimal instructions:
 - Goal: “Check it out and type gmake.”
 - It must “just works”.
- Documentation.

Documentation

- For the first few tutorials:
 - Narrative documentation is vital.
 - A reference manual alone is inadequate.
 - Start from familiar ideas and lead to unfamiliar.
 - Grand tour of the material. Then zero in on the details.
 - Don't do too much at once.
 - Teach people how to use the reference manual docs.
 - And other documentation.
- Reference manual documentation also important.
 - Will be automatically generated from source code.

Iterating the Design

- As we develop the Tutorials we learn about the infrastructure:
 - What is easy/awkward to use?
 - What is missing and needs to be there soon?
 - What is in the wrong place?
 - What is undocumentable?
 - What is unteachable?
- Apply these lessons to the next iteration of the infrastructure and update the Tutorials.
- On each iteration add more functionality.

Looking Ahead

- Long term (2007/2008)
 - Complete offline software suite for BTeV
 - Both infrastructure and physics code.
- Short term (next 6-12 months)
 - Prepare the ground for the long term.
 - Strong set of tutorials which can be used to train the staff who will meet the long term goals.

Conclusions

- We have started on the long project of developing the BTeV offline software.
- A key part of the development cycle is the creation of a suite of tutorial examples which will
 - Evolve with the project.
 - Serve as a test bed for new ideas.
 - Be part of the validation suite.
 - Be a key part of the documentation.