Run2 Joint Software Infrastructure Efforts

- Major Shared Products (yesterday)
  - RIP, Data management, Robotics support
* Class Libraries and C++ Infrastructure
  * (ZOOM)
  * Recipes and Recommendations
  * Code-technique samples
- Other Joint Software Projects (next talk)
  - Products and methodologies, ...

Yesterday you heard about major specific projects such as data handling and the Reconstruction Input Pipeline.

The next talk will discuss shared exploration of products and associated methodologies. Those are important components of joint effort.. So are the SoftRelTools efforts, which in fact are joint among Babar, CDF, and D0, and are proving useful.

I will be focussing on the creation or acquisition and support for code modules, packages and libraries -- the ZOOM effort. I'll also discuss shared support for language issues, and in the nature of gathering and organizing recipies and code samples and "how to get around these annoying problems" advice. Although this has not been part of ZOOM, since mostly the same people are involved at least atthe coordinating end, I'll group this in with the libraries mandate.

ZOOM (FPCLTF)

- What ZOOM is Doing Now
- Trends and Focus
- Relation to (new) ZOOM charge

I will present a synopsis of the activities occupying ZOOM right now and the deliverables that have been released and are in testing. Then I will briefly discuss near-term trends and changes in what we are focussing on and how development is done. Finally, I will discuss the (updated) charge of the ZOOM group, and relate our activities to the imperatives of that charge.

Current Modules

- FixedTypes and other "utilities"
  - Including portability issues
  - Will be umbrella for "coders resources"
- HepTuple
- Exceptions
- LinearAlgebra
- PhysicsVectors

The ZOOM modules being developed are determined by the January workshop and by urgent requests from CDF and D0 potential users. Some of the modules we have either released or are completing development on are:

Fixed types to provide a uniform and portable way t specify particular data sizes (FLOAT8, INT4). This opens the issue of how to provide information accessible at COMPILE-TIME to builds in the context of SoftRelTools. The current FixedTypes package is available, whithout having resolved that issue completely.

A related area which we group with this is the whole business of providing header wrappers, code snippets, frameworks, and recipes for making code that works in the spectrum of supported environments with minimal ugliness such as #defines scattered throuought a program. The ZOOM utilities repository should be a convenient accessible location for such resources.

HepTuple

- Key extensions to original --
  - Directory-style Ntuples and Histograms
  - Column-wise Ntuples
  - Improved treatment of types of Hists
  - Retrieval of data
  - Removed CLHEP dependence
- Early customer usage
  - Now ready for broader release
- To come:
  - "Lightweight" (non-HBOOK) manager using Histoscope

The HepTuple package started, of course, with the package of the same name. The immediate major need was column-wise Ntuple support, and directory structure. Our requirements include certain capabilities Babar had put in for row-wise Ntuples, because we felt the reasoning behind their needs was sound for CDF and D0 as well.

We have tried to avoid changing user interface details that would force users of the package to modify much code to move to our version.

We have some CDF users using our package and D0 is beginning to try it.

At this point we are poised to make the package availble to the community, in particular Babar if they need the extended capabilities; but we have to test it on a broader spectrum of compilers than we have done so far.

There were requests for a way to collect histograms and Ntuples without the marriage to HBOOK and Zebra, which in the long run will be support problems. This became an intent to produce a "lightweight" manager, which however started getting quite heavy. We have determined that the correct approach will be to use Histoscope as the second manager, extending Histoscope to deal with Column-wise Ntuples.

Exceptions

- Needed for other modules developed
- Semantics extends those of throw/catch
  - ZMthrow (...) instead of throw
- Allow for:
  - Ignoring exceptions, continuing in code, or throwing the C++ exception after all.
  - Hooking in handlers
  - Logging in either packaged or custom manner
- Needed for ZOOM module use
  - but available for users as well

One module which ZOOM has developed that was not explicitly requested in January is a framework for dealing with excpetions. The ability to throw an exception is a valuable development convenience, that we need to create any large ZOOM modules. But that the C++ excepetion mechanism lacks certain features that would make dealing with those exceptions more palatable for users.

Our extensions are intended to let people write a framework to survive exceptions without each user needing to try and catch around every ZOOM call.

LinearAlgebra and PhysicsVectors

- Matrix package addressing needs expressed last January
  - Available now
- PhysicsVectors extending capabilities beyond existing packages
  - Components in various coordinates; rapidity; angle decompositon; rotation/Lorentz groups; ...
- These are separate packages
  - But will have optional coordination headers (v(m), r(m), m(r), m*v, ...)

We have a linear algerbra packa that addresses the concerns about existing packages expressed last January. It is flexible in matrix dimension yet fairly optimal in working with small or specialized matrices.

We have a Physics Vectors package dealing with 3-vectors, 4-vectors, rotations, Lorentz transformations, E-B fields, and other physics concepts. This is currently in pre-release feature testing.

The LinearAlgebra and PhysicsVectors packages are kept physically distinct so that users may link in only what they need.

Near-term Focus of ZOOM

- Completion of packages in progress
- Limited number of Major packages
  - For instance extension of Histoscope + Columnwise Ntuple to use as manager for HepTuple.
* Picking, Adapting and Unifying
  - Package CLHEP parts and other freeware

The focus of ZOOM in the next short phase will migrate away from frantically providing a couple of packages needed by Run2 developers, to a three-pronged effort:

One prong is completion and polishing of the packages already released (this incudes support and mentoring early users if necessary). Another is continuing development of perhaps one major package at a time.

But the main thrust which ZOOM has not done heavily for the last few months, is a concentrated effort to identify, pick up, and package the best available freeware to meet immediate needs of our Run II users. Thus we will be going through CLHEP, for instance, and identifying which pieces stand independently of others.

We will identify which modules would need our work to be adapted or would need support locally, and of those, which ones are worth adapting and supporting. We will package and distribute the selected modules so that our users are likely to have their needs met within the controlled libraries at FNAL.

Charge to the F.P.C.L.T.F. (ZOOM)

- Basic math, stat, general physics capabilities
  - Were mostly encompassed in CERNLIB
  - Large part of CERNLIB rendered moot by language
  - Certain packages made natural by OO capabilities, which were not suitable for sharing in FORTRAN environment
  - Requirements determination is iterative:
    - Users have needed to see what ZOOM can deliver before further specifying what they need.
- Policy/Priorities/Requirements Management in hands of RUN II (computing) Steering Committee
  - Current priorities set by Jan workshop, as interpreted by that committee

There is a new proposed charge to the ZOOM effort, and for the remainder of this talk I will discuss that charge in terms of how our present efforts are meeting and/or missing those points.

The content of ZOOM was, in spirit, largely encompassed by CERNLIB in the good-old days. A large part of the FORTRAN library has become moot in a C++ environment, and the pressence of C++ makes a lot of different packages natural for re-use.

We have to have a good way of identifying where the needs lie. Thus far, the user community has been in a wait-and-see mode: Before the put the effort into defining what they need, they have to see what sort of stuff we can deliver.

The charge describes is a RUN II computing Steering Committee, consisting of CD, CDF and D0 leaders. This is not a change from the way we have been working; that committee will continue to have requirements change management responsibility and the final say in setting priorities.

OO framework:

- Callable from C++
- Encapsulated/interface-defined
- Minimal phyiscal coupling to large legacy packages
  - » Thus HepTuple is decoupled from CLHEP and (when Histoscope is in) from needing HBOOK/ZEBRA
- Support roles
  - – Packaging, releasing, distributing, feature testing each package
    - » CVS / SoftRelTools
    - » Reference repositories for CDF and D0
  - – Variety of supported platforms - Committee defines list
  - » Helping with stress testing by users

The next paragraphs in the charge describe the framework ZOOM is to work in: The mandate to shun coupling to large legacy packages where possible is worth noting.

Then a definition of support roles is given. This applies particularly to "freeware" packages, where one's responsibility does not end with saying "USE THIS".

The Steering committee will define a list of compiler/system combinations the ZOOM modules have to work under. They of course must answer to CDF and D0, and will strike a balance to avoid growing too large a list too early.

Routes for providing packages

- Evaluate and recommend commercial solution
  - Broad-spectrum exploration of available products done by users
  - ZOOM and/or users do intense investigation of --
    - Specific products seen as promising by users
    - Issues of cost, support needs, verification...
- Previously in hands of C++ working group...
  - Considerable overlap with ZOOM, including the Run II physicists guiding ZOOM directions

Finally, the charge lists three means of providing packages. The first is to recommend a commercial solution; generally, physicists want to and will do broad-spectrum investigation of products. ZOOM should do deeper investigation into how the most promising commercial solutions would be acquired and used.

Recommend, adapt, support "freeware"

- Initial import
- Official repository maintenance
  - timely updates, stability
- Mentor early users at FNAL
- Changes as needed by CDF/D0
  - Offered back to the community at large

» Specific freeware packages to be a major focus in the near future

The second approach to providing packages is to adapt freeware solutions for Run II use. I have already discussed this as being emphasized in near-future activities -- we will be picking the best of the HEP-community freeware world -- which has indeed become a lot better since a year ago.

Adapted and enhanced packages will, of course, be offered back to the genral community.

Development of new packages

- Full product cycle
  - Requirements determination
  - Design
  - Development
  - Testing
  - Distribution
  - Mentoring initial users
  - Continuing Robustness/correctness support
  - Upgrades
    - Requirements management by S.C.

To my mind the most important paragraph of the charge defines what it means to support a package. The components are listed here; except for the first three, these may apply to packages bought or borrowed as well. FOr new packages, requirements and change management is in the hands of the Steering committee. This is what will keep ZOOM responsive to actual Run II needs, and not just to the most aggressive scientists' private agendas, or for that matter, to the personal desires of the ZOOM participants.