Visualization- I

- HEPvis Summary
  - Bjorken's talk ASSERTS strongly that graphics and visualization are enabling technologies for HEP
  - 2D-3D visualization -- group favored carefully thought out 2D projections . 3D still important.
  - Debugging tool
  - may be useful for multivariate style analyses that are catching on
  - Next generation of scientists may insist on more use of visualization
  - Perhaps should consider an encore performance of HEPvis while we see what the next generation of experiments need

Visualization - II

- I am not a fun guy ... but at the risk of insulting someone:
  - A VR model is probably absolutely essential to designing and understanding a large detector
  - I would have pushed for one for SDC
  - If you've ever been to a so-called 'integration meeting' for a large detector you would understand why

- Even tho the work shown IS indeed engineering I believe that it IS also high energy physics!
- The graduate student that will work on LHC is 13, 14 years old today. We had better develop the tools that his/her generation will be comfortable with rather than those of a generation which has been sensorily deprived by limited computer hardware.

Dominance of the Commercial Sector

- Computing has always from the start been about business
- However, the scientific and research community held a special significance
- The commercial world is more dominant than ever and is driving the new technologies more than ever

- In addition, computing is now moving so fast that the part of the scientific community that focuses much of its attention on computing issues will be in the best condition to contribute in fundamental ways.
- This does not mean that we can rely on the commercial world for everything or that no significant ideas will be produced by HEP - e.g.. WWW.

- We must try to align ourselves as much as possible with the products and methods of the commercial and computing research world and reserve our efforts to produce things that do not exist or glue pieces to integrate products to solve our needs.
- The many non-HEP plenary sessions should clearly have suggested the sophisticated andintense work being done outside the narrow confines of HEP.
- We should use these methods and products to the fullest extent possible

- I was very impressed with Carl Malamud's advice to us -- not to retreat into narrowly pushing our own networking issues but to use our considerable stature to help educate and inform the public and the politicians concerning the value of networking. In doing so, we will create the impetus to increase bandwidth and services and we will derive benefits from it.

- Stu Loken was asked to emphasize non-HEP efforts in his discussion of virtual laboratories. The tour he took us on was quite impressive and represents a more ambitious goal than, for example, the remote control room initiative between KEK and FNAL. He also emphasized that pushing the use of the network would encourage the expansion of bandwidth and the creation of more efficient protocols.

Software Development

- A major purpose of OO is to control and regularize interactions between the parts of a program
- Many of its key concepts are designed to bring law and order to the chaos of computer memory space
- By doing this, the communication overhead is reduced and the coefficient of N^2 is smaller

- Since design is a major part of the process and a clean conceptual design reduces the chances of confusion, a software analysis and design methodology is necessary. This seems to be a definite part of people's plans.
- Due attention must be paid to debugging. How efficient is debugging in C++?

- What is N_{eff} ?
  - Number of people in the software group?
  - Number of fully committed experts?
  - All physicists and computing professionals who are working on some part of the analysis?
  - All of above plus computer literate kibitzers?
  - All collaborators and close families?

- Who sets requirements? Who allocates resources?
- Need a definite agreement between the collaboration as a whole and the analysis effort. Otherwise, N_{eff} is way too large.
- Need a structured software development team with clear paths and a small number of people with ultimate responsibility and control

- Obviously, two languages is BAD ..... “They are just one people, and they all have the same language. If this is what they can do as a beginning, then nothing that they resolve to do will be impossible for them”
- Of course, there is intra collaboration politics and an F90 and even an F77 constituency. If this bad thing is done, it should at least be done with a very careful plan.
- On the plus side, over the lifetime of the experiment --esp LHC -- there will be new languages so maybe it MLP is inevitable on that time scale

- I am very impressed with the progress made on using OO and C++ since the last CHEP -- MOOSE, GEANT4, talk of Victor Decyk et al
- Efforts that looked promising but uncertain can now be declared a success and many fears have been dispelled.
- However, this is a select group and will need the participation of many others in their collaborations who may not be so adventurous or resilient. Much work will have to be done on lowering the impedance to these big changes

HEPonomics

- In an era of poor prospects for permanent positions for physicists in High Energy, we must make sure that our students and RA's have training that will make them employable in the commercial sector if they choose to or have to leave HEP.
- We also have to be able to attract excellent computing professionals and they will demand to use the best tools

Simplifying the Task

- Reuse is hoped to be a way to simplify and reduce the complexity and size of software projects. It can also be viewed as ‘increasing the number of programmers’ while not increasing the communications overhead (much) .
- Components, families, and patterns all work in this direction

- Families allow one to produce a variety of customized programs in a program domain. An antidote to the humungous general purpose program. They also produce a high level language in which the users can describe their needs and therefore reduce communication overheads. And they permit automatic code generation which presumably reduces also debugging.
- Components may be viewed as reducing the size and complexity of the program or adding a programmer without adding the attendant overhead because of the well defined interface

Who should ‘do software’

- It is commonly believed in HEP that we need lots of programmers and we have to use the physicists. Is this true?
- Could we invent a rather high level language to allow the physicists to describe their problems to programmers or in some very high level manner and then get the actual software generated for them. In the early days, physicists often worked with programmers and some didn’t actually program.

- If physicists do program, should the ones who participate in fundamental architectural decisions and develop the program foundation elements be required to have rigorous training?