**General Requirements of FNAL projects on Geant4**

**LBNE (Brian Rebel).**

Brian is involved in the simulation of the Liquid Argon calorimeter. He mentions a few issues related to the physics of G4:

1. How well Geant4 handles processes like light production in LAr and low energy ionization, for example from solar or super nova neutrinos. He refers to neutrinos with E<15 MeV producing electromagnetic showers.

2. How to put together a suitable physics list, and optimize the thresholds on secondary particles. in physics lists. They have figured out how to add different processes, but they do not know how to go about adjusting parameters.

They have not yet exercised the Geant4 functionality because they are primarily concerned with just getting the software working. ArgoNeuT (Fermilab experiment exposing a small-scale liquid argon calorimeter to Numi beam) is currently starting to analyze data and will be an invaluable source of physics feedback.

**LBNE (Thomas Junk)**

Thomas is the head of offline/computing for the water Cherenkov detector. He has a concern about the modeling of hadronic showers in neutrino-nucleus interactions. A collaborator of his, Xin Qian, has recently made improvements to the Rayleigh and Mie scattering models, which are important for transporting photons properly through the water detector.

**Generic detector research and ILC related work (Adam Para)**

Adam has an observation about the release mechanism. The program requires a large number of data files; some of them are edited manually between releases. This is clearly a weak point, hard to avoid, and even harder to detect. He proposes a database-like mechanism for storing, monitoring, and retrieving the underlying data.

Adam’s main focus is the G4 physics content and its reliability. There are at least two distinct usage modes of the simulation programs: describe the performance of running detectors/experiments, and to predict the performance of some novel detection techniques or detectors. The LHC experiments are the main G4 customers and should be the focal point for the foreseeable future. The adequate description of the existing experiments does not necessarily require that all the physics processes are implemented, or that the physics models are perfect. Validation based on experimental data taken with those detectors, even by tweaking’ wrong’ knobs, would be acceptable.

The situation is quite different when using the simulation program as a predictive tool. The completeness and the correctness of the physics modeling are far more critical for this purpose. Adam is quite impressed with the spectrum of the physics processes implemented. It is a formidable repository of the knowledge we have accumulated over the past 50 years. On the other hand there are many areas, such as hadronic interactions, where the modeling is deficient or wrong. Energy non-conservation, discontinuities, unphysical behaviors are a limiting factors for Adam’s studies of in the area of high resolution calorimetry. The intrinsic G4 energy resolution is bigger than the resolution of a calorimeter he is investigating. The documentation and verification of the physics content of a simulation program is a very complicated problem and it requires major resources, intellectual and computing. Any of the future initiatives is likely to require a huge investment of resources. A reliable simulation program is a vital tool to minimize the risk. As examples, he cites the super-precise measurements associated with the FNAL program: mu2e, g-2, LBNE. The reduction of backgrounds by many orders of magnitude requires a simulation accuracy far beyond what we have at the moment. Calorimetry for future lepton colliders (ILC, CLIC, Muon cCollider) is another example. The CALICE experiment has admitted the impossibility of the experimental verification of the proposed techniques and relies on the simulation as the ultimate design tool. The US is dedicating significant resources to develop the Muon Collider technology; the environment is very challenging because of the large backgrounds. The experiments depend critically on the details of the shielding design that, in turn, depend on the correctness of the simulation programs. The latest motivates the need to increase the collaboration between the G4 and MARS teams to achieve a credible model of the nuclear effects. The documentation, verification, and validation of the physics content of G4 is primarily a scientific project and it takes knowledge and experience in various branches of physics and technology. It contains a significant computing component too; use of modern databases and high-end graphics can make the task of comparing and documenting easier.

**mu2e simulation (Robert Kutschke).**

Mu2e simulation is at an early stage. They are learning how to use G4 wisely and where its limitations are. Mu2e will eventually need to prove to a review committee that G4's hadronic and EM models are accurate enough for the purpuses of the experiment. Rob thinks they would benefit from validating the simulation with thin target data and their own test beam data; they are willing to work on these topics although they acknowledge no experience. Many people in Mu2e are concerned that preliminary results from HARP are quite different than previous results and they would want to study and incorporate them to G4. They also need to understand neutron transport throughout the experimental hall, pion production from the primary interaction of

protons on the production target, particle production from the nuclear interaction of pions with the stopping target, capture of muons on several different materials.

The capture of muons is really a collection of 4 processes:

1) The formation of a muonic atom.

2) The decay of a muonic atom; this includes decay in orbit, capture

 on the nucleus and mu to e conversion.

**g-2 simulation (Kevin Lynch).**

Kevin mentions that it would be nice for G4 to handle hardware parallelism as opposed to “running more jobs” .

**Some questions, suggestions, issues by Fermilab Users**

**1 Mu2e (contributed by Rob Kutschke)**

1) G4Runmanager:

 - BeamOn is split into 4 methods.

 - There is no change in functionality, just a refactoring.

 - This is done to control the event loop from our framework.

 - Need to propose to G4 that they adopt our refactoring. How to do this?

2) G4Runmanager:

 - They have an elegant solution to the management of the lifetime of the many objects

 that one must "new" during the construction of a G4 geometry. This is a nice method

 to make sure that all of the deletes get called at the right time.

 - Need to propose that G4 adopts their solution, or provide with another solution.

3) Hadronic physics. When protons hit the production target, they would like to run a

 special process that lets them sculpt the distribution of produced pions. Once this is

 done, they would like G4 to treat protons as it would without their change. They only

 want this behaviour in the production target. Is it possible to do this with the existing

 toolkit? If so, how? If not, they would like to request this as a new feature.

4) The 4 processes that make up muon capture on a nucleus are currently implemented as

 a single G4 process that does it all. With this implementation, it is awkward to tweak

 parameters of the processes. A mu2e collaborator is writing code that will implement

 the above physics as 4 separate G4 process, each of which can be configured

 independently of the others. They would like to contribute this code?

5) Mu2e is using the QGSP\_BERT or QGSP\_BERT\_HP physics lists. In order to speed

 up the simulation they have implemented UserTrackingAction, UserStackingAction

 and UserSteppingAction code that kills particles for a variety of reasons. Probably

 there are supported ways to do this by setting cuts in the physics processes. They have

 done this their way since they don't understand possible side effects of setting cuts.

 They would like to work with someone to learn how to do this correctly. For example,

 they would like to set minimum kinetic energy cuts ( or possibly range cuts if that is

 preferred ) as a function of physical volume and particle species. They also want a

 clean way to turn this on and off.

6) G4 graphics is used in a fairly primitive fashion. They would like to understand how to

 do much better graphics without having to invest months to try out all of the available

 products

7) In user Stepping Action how does one know if a particular step is the first step on a

 track or that last step on a track?

8) How can the stored trajectory information be accessed?

**2 G-2 and Mu2e (contributed by Kevin Lynch)**

The most important issues are field, visualization, and particle related. There are various mixtures of E&B fields all over the simulation, but in many places, they also have time dependent E&B fields. It would be very useful to have

1) Visualization of fields

2) Visualization of time dependent fields

They find the options for visualizing complicated geometries deficient. All the standard visualizers have serious bugs that they need to tickle with their code. These are typically of three forms:

1) They can display everything, but are so slow (HepRep in JAS) or user unfriendly (DAWN), that they can’t realistically be used.

2) They can't display all the geometry (the OGL viewers don't display various parts of the g-2 geometry, depending on the instantiation order), or

3) They display the geometry incorrectly (the OGL viewer has serious problems with Polycones).

The next issue is not exactly a requirement for g-2, but is for Mu2e and could be somewhat useful for g-2: defining new particle types is more difficult than it should be, due to various significant design flaws in the particle/process code hierarchy. There are two main issues: runtime particle creation and inheritance of process implementations.

1) The most significant problem is that one can only instantiate new particle types when the state machine is in the PreInit phase. It can be done outside PreInit, but not without jumping through significant hoops.

2) The explicit checks that processes make on the particles that they are called for are often done by particle name rather than using the type system where appropriate. This implies a "cut-and-paste inheritance" of process code for derived class types, rather than forward compatible, compiler enforced, inheritance.

3) Processes are given static ProcessType and ProcessSubType, rather than a dynamic registration mechanism. This means that, when writing process code that doesn't live in the Geant4 distribution, the user can never be sure that the chosen process type integers won't conflict with future upgrades to Geant4. Because of this, it is not clear to that these process types have any portable use at all.

An additional problem is the poor quality of the documentation, in particular the part related to the structure and behavior of physics processes. The advice from G4 is not to “do it yourself by use the builders”. But the builders are undocumented.