Summary of Session 6A Re-thinking the Hadronic Framework

Dennis Wright

14 September 2012

Why Change the Hadronic Framework?

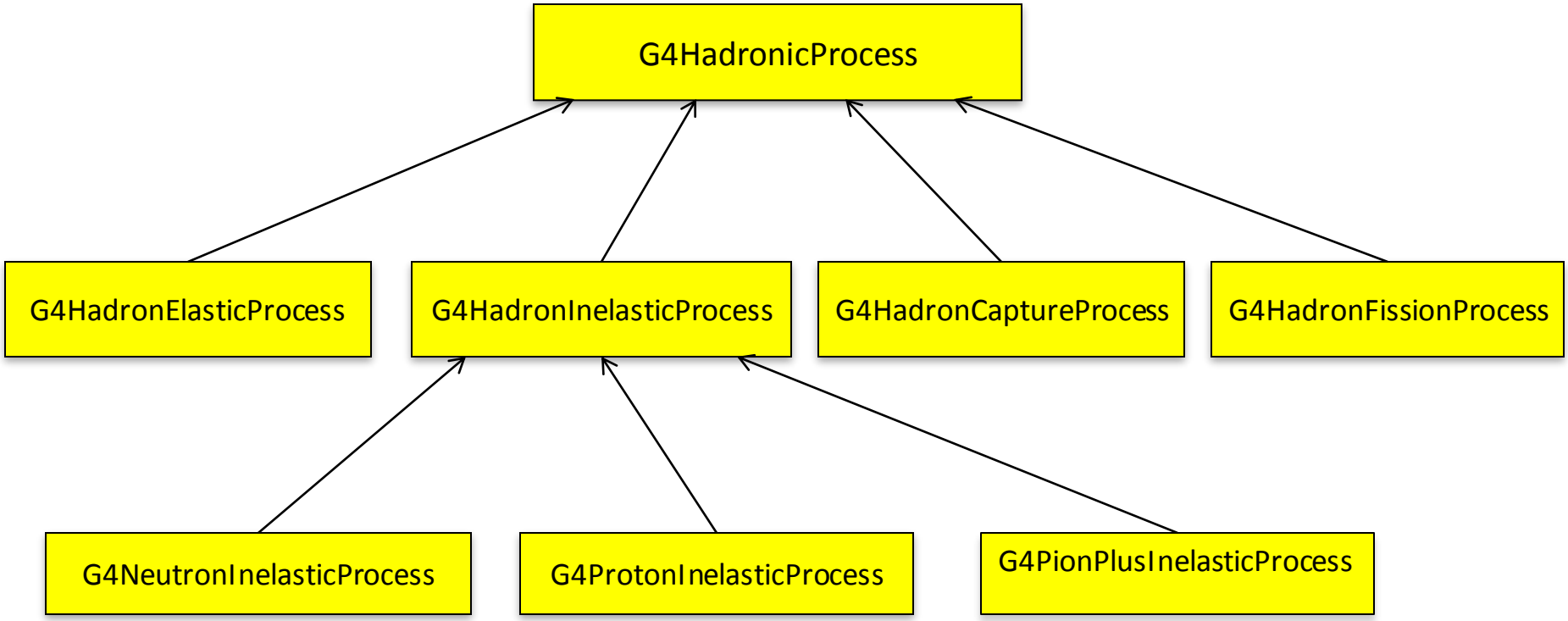
- It may not be flexible enough
  - does not easily accommodate all hadronic processes, models and cross sections
- It may be too deep
  - too many levels of inheritance which complicate and slow down code
- It may give too much (or not enough) control to users
  - should we consider more defaults (other than cross sections)?
- Could be extended to include nuclear physics applications

Removing Inheritance Layers

- Removing inheritance layers MAY
  - significantly reduce execution time
  - improve multi-threaded, multi-CPU behavior

- Remove layers in hadronic process inheritance
  - remove G4HadronInelasticProcess (doesn't do much)
  - remove G4ProtonInelasticProcess, …. (replace with template?)
- Remove layers in model inheritance
  - remove, e.g., G4VPartonStringModel
  - maybe others

Current Inheritance Scheme for In-flight Hadronic Processes



Removing Inheritance Layers: Conclusions

- Removing layers in hadronic process inheritance will be useful and worth a try IF
  - speed advantage can be demonstrated
- Not clear if removing layers in model inheritance is worthwhile
  - more study required

Do We Need Capture and Fission Processes?

- Some models now handle capture automatically
  - -> make capture a part of in-flight process

- Fission is already part of several in-flight models
  - G4LFission (GHEISHA) only active, stand-alone fission model
- No conclusion whether to remove or not

At-rest Processes

- A consistent treatment of in-flight and stopping processes is desired
  - G4HadronicProcess (derived from G4VDiscreteProcess) is the base class of most hadronic processes
  - many stopping processes derived from G4VRestProcess instead
- In current system:
  - stopping and in-flight processes can never be treated equally
  - stopping processes cannot inherit useful methods from G4HadronicProcess or use model approach of framework
- Conclusion:
  - Make G4VRestDiscreteProcess base class of G4HadronicProcess
  - move process sub-type enum to ctor of G4HadronicProcess
  - use sub-type flag to decide whether in-flight or at-rest methods should be used

At-rest Processes: Additional Proposal

- We could also derive G4HadronicProcess from G4VProcess
  - removes an additional layer in hierarchy
  - need to consult Processes Category developers
  - in meantime, will proceed with derivation from G4VRestDiscreteProcess

HP Neutrons

- HP and LEND models require material pointers and do their own sampling of isotopes
  - this adds a lot of complication to G4HadronicProcess and to cross section classes
  - such complication is not required for any other model
  - specialized inheritance for HP and LEND
  - possible inheritance diagram on slide 5
- G4MaterialDependentNeutronProcess
  - would have G4Material pointer
  - other processes would not have material pointer
  - would do its own isotope selection
- No conclusion – more study required

Cross Section Review and Clean-up

- Cross section classes still not handled clearly or consistently
- Re-design completed more than a year ago
  - some planned migrations completed, not all
  - end result not very satisfactory
  - one reason: material dependence of HP neutron models
- Factory-based mechanism to assure a single instantiation of a cross section which may be used by more than one different entity
- General means for smooth blending of one cross section set into another vs. energy
- Agreed to pursue clean-up, but depends on decision to split out material dependence
- Smooth blending requires more study to achieve general solution

Framework Rules

- Currently have default cross sections but not default models
  - add default models?
  - no conclusion

- G4HadFinalState
  - currently must copy into particle change
  - can we modify particle change or G4HadFinalState classes to avoid this copying?
  - more study required

- User hooks into hadronic models – good idea or bad idea?
  - agreed to provide standard user interface to models
  - administrative controls to decide which parameters to be accessible

Nuclear Physics Extensions

- Want user-selected explicit final states (n,2n), (p,n,π^+), etc.
  - easy to do as biasing option on top of cascade models
- Want user access to nuclear target
  - difficult for NeutronHP, continue to work on problem
- Provide "supermodel" to choose appropriate cascade model for ion-ion collisions based on A of target and projectile
  - some study required
- Provide J vectors for initial ground and excited nuclear states in G4Nucleus, G4Fragment
  - can do

Backup Slides

-

Model Inheritance