Factorization of Common Code for the Hadronic Tests

Julia Yarba
Fermilab

18th Geant4 Collaboration Workshop
09/23/2013

Disclaimer

- Ideas started out from practical consideration (not physics), such as code maintenance efficiency, etc.
- Preliminary thoughts presented at the Geant4 HAD group meeting in July 2013, and were initially supported
- One of the questions was about introducing changes into existing and operational Geant4 testing machinery
- Some aspects of our initial discussion will be revisited
- Several possible scenarios for further steps will be considered
- Common code can be possibly re-used in newly-introduced packages, to reduce the amount of work
- Changes in existing packages are NOT mandated
- Feedback will be greatly appreciated

Motivation

- Lots of coding similarities across a number of applications in the Geant4 Hadronic Validation domain
- Apparently, it was influenced by one particular test (test30 ?)
  - Which mean it was an excellent example
- There is non-negligible code duplication across the suite
- Note: I've got several such tests, and I'd like to simplify the structure - don't call me "selfish" !
- Factoring out some of the common-use SW elements into a separate library/package is likely to be a benefit

SW Pieces in G4 Validation Tests (I)

- Geometry
  - "Virtual" (material+"G4 req.minimum" – for model/process-level tests)
  - Realistic (for physics lists tests)
- Physics
  - Process level (single interactions)
  - Physics Lists (combination of models, with overlaps in validity range)
- Beam definition – particle type, kinematics
  - G4Track with Pre/PostStep points defined (model/process-level tests)
  - G4VPrimaryGeneratorAction (physics lists level tests)
- Run Control
  - G4ProcessManager – allows for loops over different beam/target/model combinations in the same job
  - G4RunManager

SW Pieces in G4 Validation Tests (II)

- Configuration/Steering
- RNDM engine management, revisions for parallel processing
- Misc. (user actions, such as stepping, etc.)
- Plots/Results:
  - Observables – greatly determined by exp.data, test specific
  - SW to access and plot simulated observables
  - Exp.datasets, in what format to store (currently ASCII)
  - Analysis/Display scripts

Initial Work (so far)

- Currently, the 1st try is test23/CommonSW (in SVN/trunk)
  - At present, builds as a separate library
- Test23 (phys.list) and test19 have been adapted to use it
- Would like to adapt test47, test48, test75
- BUT !!!
  - Use of common library means additional dependency
  - Needs to be smooth transition for those in CTest
- Calls for a technicality, i.e. CMakeList
  - Triggers build of the common lib if needed (or equivalent act)
  - Remembers the location of the common lib
  - Triggers rebuild of tests in a change in common code is made
- OR AN ALTERNATIVE SCENARIO ?

Possible Scenarios for Further Steps

- Code Name ????
- Package location (in the repository)
  - geant4/tests/test23/CommonSW
  - geant4/tests/CommonSW
  - Other suggestions ?
- Package use/build
  - Library
    - But currently implemented Cmake scripts do not trigger library rebuild if change is made in the common code
  - Similar to geant4/examples/extended
    - common code is explicitly compiled with each example application
- Other suggestions ?

Geant 4

(Instead of) Summary

- In principle, introduction of a common-use code in the (hadronic) validation domain may simplify maintenance
- May also reduce implementation overhead in new packages
- It should blend smoothly with existing testing procedure
- Decisions need to be made:
  - Package Name
  - Package Location
  - Package use – library vs direct compilation with each test
- Feedback/review/suggestions are most welcome