Simulation Framework Subproject

http://lcgapp.cern.ch/project/simu/framework/

Witek Pokorski

31.03.2005

Outline

- Goals of the subproject
- Current activities
- Future plans
- Conclusion

General goal of the subproject

- to provide flexible infrastructure for the development, validation and usage of Monte Carlo simulation applications
  - provide means of interaction between experiments and simulation toolkits developers
  - provide reusable tools/elements eliminating duplication of work and divergence
  - support physics validation subproject

Specific workpackages

- Geometry Description Markup Language (GDML) - R. Chytracek, W. Pokorski
- FLUGG applications/extensions - W. Pokorski, A. Ribon
- Python interface to Geant4 - W. Pokorski
- Geometry persistency for G4 (in the plans) - W. Pokorski, …
- Universal Monte Carlo truth handling (in the plans) - W. Pokorski, …
- VirtualMC

GDML

- initial frame: XML based geometry description for Geant4
- proposed by 'Geometry Description' RTAG, became 'LCG-supported' project in October 2003
  - motivation:
    - move away from hardcoded geometry
    - enable geometry interchange between different applications
    - no direct binding across applications
- presently part of the Simulation Framework subproject (Simulation Project) as the geometry interchange format workpackage
- needed to be able to exchange (providing **universal** exchange format) geometries between different applications
    - between different Geant4 applications
    - between different simulation engines and/or reconstruction programs

GDML - present status

- GDMLSchema (set of XML Schema Definition files) supports realistic geometries such as LHCb or ATLAS
- GDML writer working and tested on full LHCb geometry and on several ATLAS subdetectors
- GDML readers exist for Geant4 (C++ and Python implementations) and for ROOT (Python implementation)
  - Geant4 -> GDML -> ROOT successfully tested on LHCb geometry
- major package reorganization undertaken recently
  - less coupling between GDMLSchema and GDML processors
    - allows easier development of GDML processor
  - improved build system to allow better portability

GDML - plans

- continue testing on realistic geometries (ATLAS, CMS)
- implement missing elements requested by users (divisions, reflections, advanced parameterizations)
- support, support, support...

- introduce modularization of GDML files

FLUGG applications

- underlying idea
  - perform Geant4 - Fluka testbeam simulation validation
- approach
  - use FLUGG to call Geant4 geometry from FLUKA
  - implement FORTRAN - C++ interface to create OO hits/digits
- first example application for Atlas Pixel Detector by A. Ribon
- note in preparation
- another concrete application (ATLAS TileCal) to be implemented in collaboration with Physics Validation subproject
- general example could be added to FLUGG distribution

Python interface to Geant4

- almost all LHC experiments (and not only) expressed interest in Python (LHCb, ATLAS, CMS)
  - some started their own implementations
- Geant4 planning to provide support for Python interfaces for steering the simulation
- the goal is to come up with some common solution
- SEAL provides tools Reflex/PyReflex perfect for producing Python binding for C++ classes
  - fully non-intrusive reflection
  - fully non-intrusive Python binding
- simple Python-driven G4 example has been implemented, demonstrating the usage of those tools
- Python is perfect for playing the role of 'glue' between applications

Python G4-ROOT Example

- initialize Geant4 from Python (using PyReflex)
- load GDML geometry into Geant4 (using GDML->G4 Python binding)
- initialize ROOT from Python (using PyROOT)
- load GDML geometry into ROOT (using GDML->ROOT Python binding)
- run Geant4 simulation from the Python prompt
- visualize geometry and hits using ROOT
- plot histograms/perform preliminary analysis using ROOT

- perfect for quick development/debugging
- allows easy use of existing tools
  - reduces the need to develop sophisticated frameworks

New items to be studied (1/3)

- G4 Geometry persistency using POOL/ROOT
  - Geant4 does not come with any concrete geometry persistency mechanism
    - could be useful for quick storing/retrieving geometry
  - POOL (ROOT I/O) provides machinery to store C++ objects
  - Geant4 could use it for (almost) free
    - some technical problems need to be solved

New items to be studied (2/3)

- Monte Carlo truth handling
  - not trivial problem due to large amount of particles (for instance in showers)
  - selection criteria (what to keep) often complicated and requiring some 'buffering'
  - Geant4 proposes 'trajectory containers' to store MCTruth
    - not very flexible
  - several experiments came up with their own (not always the most efficient) solutions
  - some 'light-weight', flexible common tool/framework would certainly be useful

New items to be studied (3/3)

- VirtualMC
  - certainly interesting approach, especially for physics validation
    - waiting for Fluka binding to be available
  - closer collaboration with Simulation Framework subproject will certainly be (mutually) beneficial

People involved

W. Pokorski (subproject leader)	0.5 + 0.2(GDML maintenance)
R. Chytracek	0.1 (GDML)
A. Ribon	~0.1 common effort with Physics Validation subproject (FLUGG)

- **in total less than one FTE...**
- **will request a technical student to work on GDML**

Milestones for 2005

GDML package reorganization (allowing independent releases of GDMLSchema and GDML processors); new release of GDML	end of March
Working example of Python-driven Geant4 application (using Reflex and PyReflex)	end of March
GDMLSchema extension to support divisions and reflections	end of June
Feasibility study of POOL/ROOT based Geant4 geometry persistency	end of June
Python implementation of GDML writers for ROOT and Geant4	end of June
Introduction of modularization of GDML files allowing selective importation of parts of the geometry	second half of 2005
(common milestone with the Physics Validation subproject) FLUGG application to one of the LHC calorimeter testbeam simulation; documentation of general FLUGG usage in the context of testbeam validations	second half of 2005
Proposal for universal handling of Monte Carlo truth	second half of 2005
Evaluation of VirtualMC for usage in Geant4-FLUKA testbeam validation studies	second half of 2005

Conclusions

- the subproject is delivering tools facilitating development and validation of simulation applications
- it is playing the role (we believe, essential) of an interface between LCG software (SEAL, ROOT, POOL, GDML, ...) and simulation toolkits
  - it is applying other LCG products in the context of simulation
- recently most of the effort has been put into GDML and Python interfaces
  - large interest expressed from LHC experiments and also groups outside LCG
- it is supporting the physics validation subproject within the limited manpower available
- there is a number of areas where Simulation Framework subproject can contribute during LCG phase II