Integration into Geant4

Memoization of Cross Sections

P. Ruth, P. Diniz, R. Fowler, A. Dotti

renci

USC University of Southern California

U.S. DEPARTMENT OF ENERGY
Office of Science

SLAC NATIONAL ACCELERATOR LABORATORY

Content

- Paul presented the algorithm and the benefit for Geant4 applications in previous talk

- Here concentrate only on:
  - **Integration with Geant4 toolkit**
- Hadronic Working Group has discussed and agreed on trying this out during 2015 to be fully released to users in Geant4 Version 10.2 (Dec. 2015)
- Details presented here have not been fully discussed with WG, so we may need to iterate if we have misinterpreted something (e.g. interaction with specialized codes like neutron HP)

Design

General Design

- Reminder (simplified):
  - Geant4 propagates one **`G4Track`** at the time: four-momentum + particle definition (mass, charge, …)
  - Attached to the **`G4ParticleDefinition`** there is a list of processes valid for that particular particle (e.g. ionization, decay, hadronic interaction)
  - **`G4HadronicProcess`** is the interface for any hadronic interaction
    - It holds an instance of **`G4CrossSectionDataStore`** : cross-section calculation interface
  - **`G4CrossSectionDataStore`** holds one or more **`G4VCrossSectionDataSet`** (base class for concrete implementation of **σ** calculations)

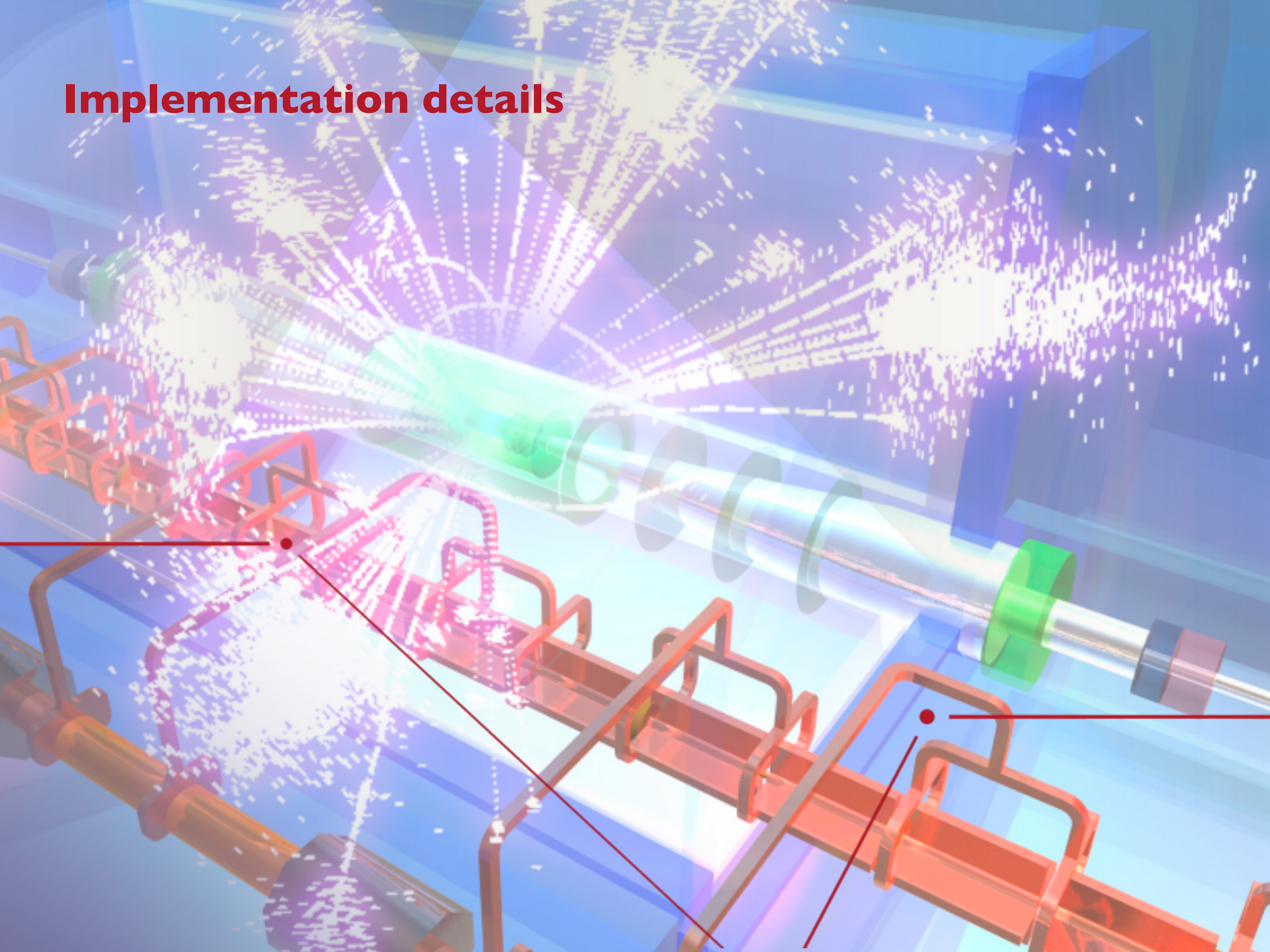
- When cross-section is needed: kernel asks each **`G4HadronicProcess`** that delegates **`G4CrossSectionDataStore`**, that asks **`G4VCrossSectionDataSet`**

- Given a process type: input of algorithm is {particle,E,material} ; output is double

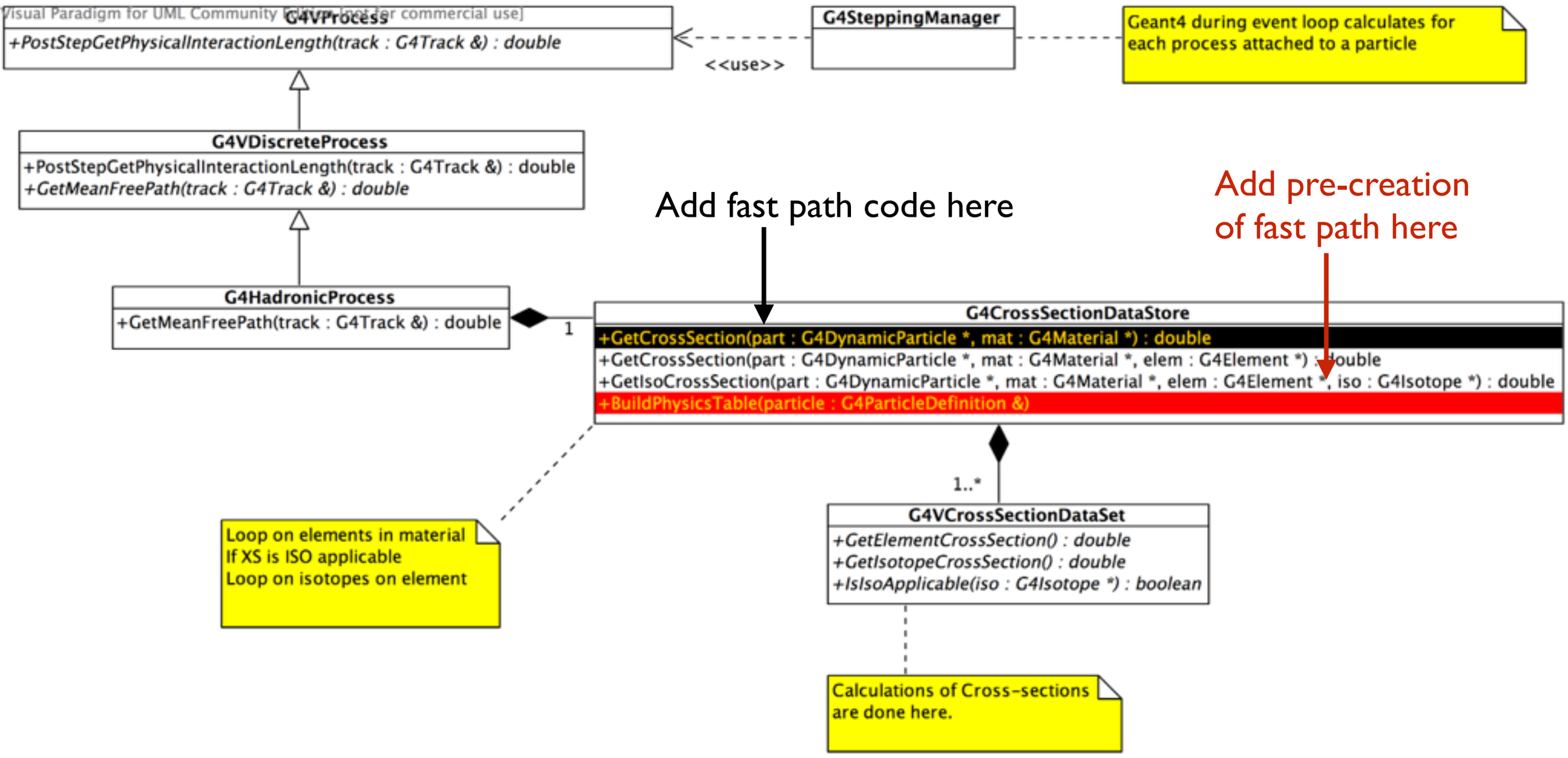
Speed-up

- Modify `G4CrossSectionDataStore` to hold a simplified representation ("histogram") of σ
  - General and independent of concrete implementation of `G4VCrossSectionDataSet`
- In some cases small **degradation of physics precision** (simplified σ) and small **increase in memory use**
  - Allow for user to selectively activate this feature
  - Done for the triplet `{particle-type, process-type, material}` gives maximum flexibility (e.g. activate only for n-elastic in absorber of calorimeter)
- **Application domains may have different requirements (keep it optional)**
  - While HEP may be ok with some rare processes to be approximated (e.g. CMS already employs Russian roulette biasing for n in calorimeters), Shielding applications will probably never approximate neutron transport

Implementation details



Classes involved



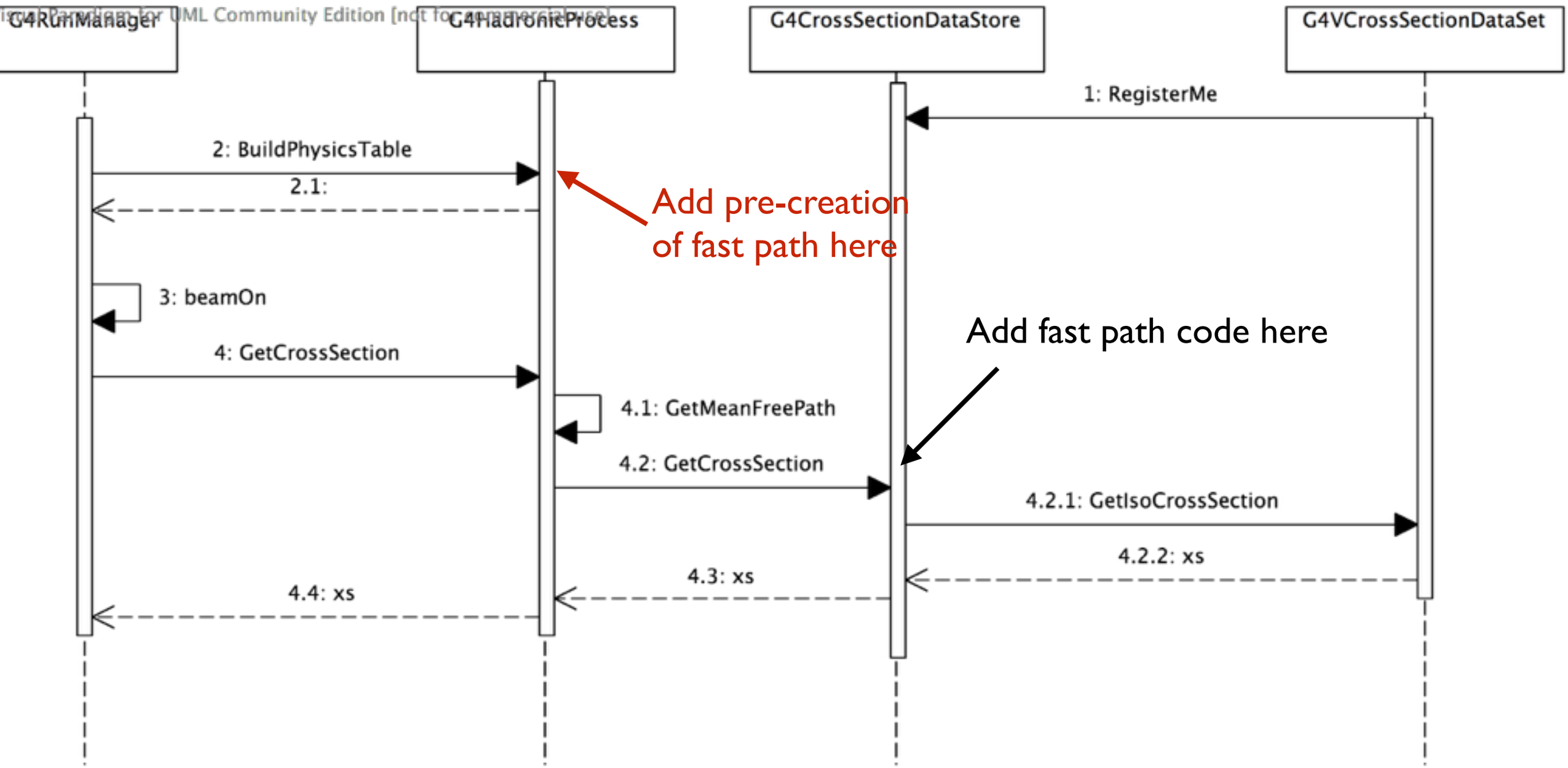
Tables initialization

- Geant4 has two states that are relevant for us:
  - `G4State_Idle` : Physics and geometry are initialized, ready to go
  - `G4State_EventProc` : An event is being processed

- During transition to `G4State_Idle` processes are signaled:
  - EM processes build physics tables
  - HAD processes usually do nothing (because **σ** are in general computed during the event loop when needed)
  - Modify `G4HadronicProcess` similarly to em ones to build simplified path data structures
    - Only for selected processes

- During `G4State_EventProc` state, when appropriate, use simplified path for HAD **σ** calculations

Sequence diagram



UI and C++ APIs

- Users will have final responsibility for activation:
  - Provide both C++ APIs and UI commands
- `G4HadronicProcess::ActivateFastCrossSection( G4Material* mat = 0, G4ParticleDefinition* pd=0);`
  - `material == 0` : apply to all materials
  - In general no need to specify **`pd`**, each process is associated to a given particle, maybe useful for future "particle shared" processes
- Corresponding UI command:
  - `/process/had/fastCrossSection <particleName> <procName> <materialName>`

MT considerations

- Reminder: particle definitions are shared among threads, but each thread has its own list of processes (to avoid mutexes)
  - Thus `G4CrossSectionDataStore` is thread-private
    - **Fast path can have thread-variant data**
- G4 MT design: master thread can perform operations and then threads can refer to it for thread-invariant data structures
  - E.g. EM cross-sections data tables and geometry
- **Allow for thread-shared data structure containing approximated cross-section** (memory saving)
  - Filled once at initialization by master thread
  - Accessed during event loop in "read-only" mode by all worker threads

Conclusions

Speeding up hadronic cross-sections

SLAC

- Proposed algorithm can be integrated into current hadronic framework
  - Need to add **only a single new public method** to G4HadronicProcess
- **By default it is turned off**
  - Because it introduces approximations and increase in memory use
  - Application dependent needs
  - Can be reviewed in the future
- Users can activate fast path via:
  - C++ API
  - UI command
- **Validation**: relatively simple, for example with SimplifiedCaloriemter run twice same setup w/ and w/o UI command in macro

Possible work plan

- End of developments: ~**now** (Paul + All)
- Starting implementation in G4 framework: **by end March** (Andrea + All + HadWG)
- Validation (physics and technical): **in time for 10.2.beta** (Validation team)
- Further work (after 10.2.beta): how to treat errors?
  - For theory driven cross-sections, what is the systematic error associated?
  - For data driven cross-sections (e.g. fits), what is the statistical (+systematic) error?
  - How to treat these in the memoization approximations?
    - BTW: this is a problem of G4 **σ** themselves, not really of this algorithm…