EM Model Catalog

V. Ivanchenko

CERN & Tomsk State University

17 September 2021

Outline

- Motivation
- Current implementation
- Discussion on indexes and names
- Plan

Motivations

- Since 2008 we have process subtype (G4VProcess::GetSubType())
  - Simple enumerator G4EmProcessSubType.hh ID = 1-25
  - Optical processes G4OpProcessSubType.hh ID = 31-36
  - G4DNAModelSubType.hh ID = 0-5
  - Stable ID for each process
  - Separate enumerators for EM, DNA, and hadronics models without overlaps
  - Used independently for different applications
- There are a need to identify model type
  - There is an implementation of Makoto which is dynamic
    - in different applications IDs may be different
    - the same IDs may be in EM and hadronics
    - Was working in TestEm5 and other applications
  - After 11.0beta Alberto Ribon introduced new system which is proposed for 11.0
    - In the next slides there are copies of slides from hadronic meeting
- We have to accept this variant or propose modifications to 11.0
  - Model IDs and name should be frozen in 11.0
  - We should come to agreement during this workshop

A. Ribon talk at Hadronic meeting

Creator Model ID (1/4)

- Unique identifier of the physics model (EM, HAD, *etc*.) responsible for the creation of a track
  - The solution in the G4 10 series is based on the class ***G4PhysicsModelCatalog*** that does the job, but many users would prefer to have a **fixed-forever model identifier**, – *i.e.* which remains the same for all applications, physics lists and G4 versions
  - The major release G4 11.0 offers the possibility to change it, according to the following solution, discussed and agreed last year:
    - **3 redundant**, useful information to identify uniquely the model creator
      - **ID** : a (large & sparse) integer that provides the type of physics (useful for analysis)
      - **Name** : a string with the name of the model (useful for debugging)
      - **Index** : a (small, contiguous) integer: the "index" of the vector of IDs & Names (useful for plotting)

4

Creator Model ID (2/4)

- First implementation in G4 10.7.ref**07**
  - Please take a look: there is still time to change some convention on ID values or Names given to models
- After G4 11.0, **ID**s and **Name**s of existing models will be frozen
  - Because users want to have fixed-forever identifiers
- New models introduced after G4 11.0 need :
  - To have their **ID** and **Name** to be included at the bottom of the method ***G4PhysicsModelCatalog::Initialize()*** in the file ***global/management/src/G4PhysicsModelCatalog.cc***
  - **ID** and **Name** should follow some convention – clearly specified as comments in the above method
    - **Index** is for free, as the index of the ID and Name vectors corresponding to the position where the model has been added

Creator Model ID (3/4)

- Here are the 3 methods of *G4Track* to get the model identifiers:

  *G4int G4Track::**GetCreatorModelID**()*
  *G4String G4Track::**GetCreatorModelName**()*
  *G4int G4Track::**GetCreatorModelIndex**()*

  Note: there is only one *Set* method:
  *void G4Track::**SetCreatorModelID**(const G4int id)*

- Inside Geant4 code, only the ID and Name should be used
- Users should primarily use the ID, and eventually Name; Index can be useful only for plotting
  - Because it has a small and contiguous set of values

Creator Model ID (4/4)

ID convention

- 1000 – 1999 : EM models
  2000 – 8999 : HAD models
  9000 – 9999 : Other (non-EM and non-HAD, *e.g.* biasing, *etc.*) models
- 2000 – 2999 : gamma- , lepto- , neutrino-nuclear models
  3000 – 3999 : elastic, charge-exchange, quasi-elastic, specialized diffraction
  4000 – 4999 : high-energy hadronic models (*i.e.* string models)
  5000 – 5999 : intermediate energy models (*e.g.* cascade, QMD, *etc.*)
  6000 – 6999 : pre-equilibrium/de-excitation
  7000 – 7999 : low-energy data driven (*e.g.* ParticleHP, LEND, Radioactive Decay)
  8000 – 8999 : others hadronic models (*e.g.* stopping, fission, coalescence, *etc.*)
- Example: BERT

```
// Class: G4CascadeInterface
theVectorOfModelIDs->push_back( 5000 );
theVectorOfModelNames->push_back( "model_BertiniCascade" );
```

```
class G4PhysicsModelCatalog {
  public:
     ...
     static void Initialize();
     static const G4String GetModelNameFromID( const G4int modelID );
     static const G4String GetModelNameFromIndex( const G4int modelIndex );
     static G4int GetModelID( const G4int modelIndex );
     static G4int GetModelID( const G4String& modelName );
     static G4int GetModelIndex( const G4int modelID );
     static G4int GetModelIndex( const G4String& modelName );
     static G4int Entries();
  private:
     ...
     static void SanityCheck();
     static std::vector< G4int >*      theVectorOfModelIDs;
     static std::vector< G4String >* theVectorOfModelNames;
};

void G4VUserPhysicsList::Construct() {
    if ( G4Threading::IsMasterThread() ) G4PhysicsModelCatalog::Initialize();
    ...
}
```

G4PhysicsModelCatalog::Initialise()

```
// Class: G4eIonisation
  theVectorOfModelIDs->push_back( 1010 );
  theVectorOfModelNames->push_back( "model_eIoni" );
  theVectorOfModelIDs->push_back( 1011 );
  theVectorOfModelNames->push_back( "model_eIoni_split" );
  // Class: G4MuIonisation
  theVectorOfModelIDs->push_back( 1020 );
  theVectorOfModelNames->push_back( "model_muIoni" );
  theVectorOfModelIDs->push_back( 1021 );
  theVectorOfModelNames->push_back( "model_muIoni_split" );
  // Class: G4hIonisation
  theVectorOfModelIDs->push_back( 1030 );
  theVectorOfModelNames->push_back( "model_hIoni" );
  theVectorOfModelIDs->push_back( 1031 );
  theVectorOfModelNames->push_back( "model_hIoni_split" );
  // Class: G4ionIonisation
  theVectorOfModelIDs->push_back( 1040 );
  theVectorOfModelNames->push_back( "model_ionIoni" );
  theVectorOfModelIDs->push_back( 1041 );
  theVectorOfModelNames->push_back( "model_ionIoni_split" );
```

```
// Class: G4alphaIonisation
theVectorOfModelIDs->push_back( 1050 );
  theVectorOfModelNames->push_back( "model_alphaIoni" );
  theVectorOfModelIDs->push_back( 1051 );
  theVectorOfModelNames->push_back( "model_alphaIoni_split" );
  // Class: G4hhIonisation
  theVectorOfModelIDs->push_back( 1060 );
  theVectorOfModelNames->push_back( "model_hhIoni" );
  theVectorOfModelIDs->push_back( 1061 );
  theVectorOfModelNames->push_back( "model_hhIoni_split" );
  // Class: G4mplIonisation
  theVectorOfModelIDs->push_back( 1070 );
  theVectorOfModelNames->push_back( "model_mplIoni" );
  theVectorOfModelIDs->push_back( 1071 );
  theVectorOfModelNames->push_back( "model_mplIoni_split" );
  // Class: G4PolarizedIonisation
  theVectorOfModelIDs->push_back( 1080 );
  theVectorOfModelNames->push_back( "model_pol-eIoni" );
  theVectorOfModelIDs->push_back( 1081 );
  theVectorOfModelNames->push_back( "model_pol-eIoni_split" )
```

Discussion on indexes and names

My Comments

- G4VProcess::GetSubType() is the main ID for analysis of the final state
- What is the purpose of model ID?
  - Distinguish secondary e- : is it delta-electron, Auger e-, PIXE, or triplet e-
  - Distinguish secondary gamma : Is it bremsstrahlung, fluorescence, triplet gamma, or bremsstrahlung splitting
  - We must freeze the scheme during G4 workshop
- Possible improvements:
  - ID numbers should be optimized for various tasks and domains
  - We should not try to have a unique ID per model
    - In the current scheme we have interval 1000-1999 for EM
  - We should foresee possibility to addition of new IDs
  - When we add a new model, developers should not edit of G4PhysicsModelCatalog – it should be possible to peak up some default ID for a given kind of secondary

Plan

- We need to decide numbering scheme
  - What interval of Ids to use
  - How to organize
- There are methods in G4Track:
  - SetCreatorModelID(const G4int)
  - GetCreatorModelID() const
  - GetCreatorModelIndex() const
  - GetCreatorModelName() const
  - We use these methods without any change
- There is G4PhysicsModelCatalog
  - We may propose modification for numbering scheme
  - All other modifications only in EM libraries and examples
- I will try to create G4EmModelID.hh and post it for discussion during weekend
- Please, be active and comment before Wednesday