Updates to Analysis

I. Hrivnacova
IJCLab Orsay (CNRS/IN2P3)

29th Geant4 Collaboration Meeting, Catania,
7 October 2024

Outline

- Updates in g4tools
  - No updates in g4tools related to analysis
- Updates in analysis
  - Overview of developments in 11.2
  - New developments in accumulables for 11.3
- List of fixes
- 2024 Work plan items

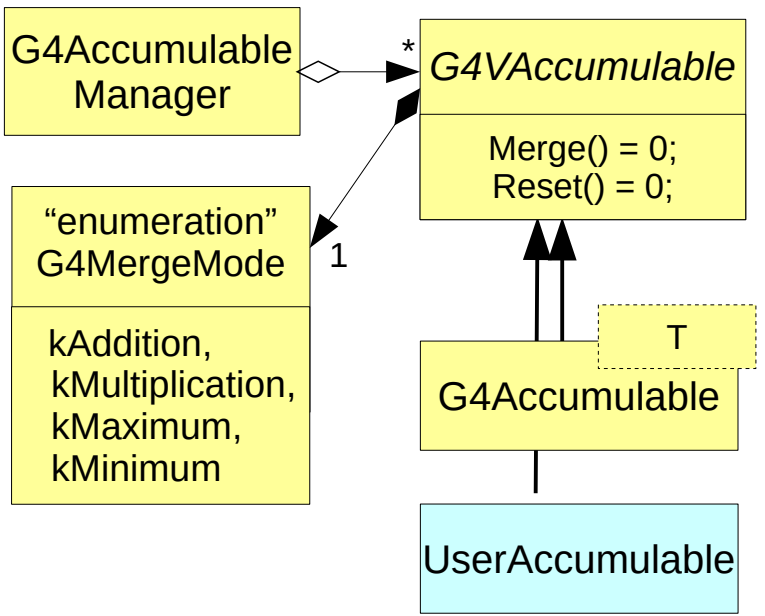
Developments in 11.2

- Added G4AnalysisManager `Delete` functions for all analysis objects types (H1, H2, H3, P1, P2, Ntuple) and corresponding UI commands
  - Requested several times at the G4 User forum
- Added G4GenericAnalysisManager `GetNtuple` function which returns ntuple_booking, that is common to all output types
  - Different from the output specific managers where the output specific ntuple type is returned
- Added UI commands for creating ntuple
- New extended example **analysis/AnaEx03 and new test320**
  - Usage of analysis commands for file management, in particular writing histograms and ntuples in a file multiple times and then also histogram deleting & re-creating via UI commands
- Presented in more detail in the last CM

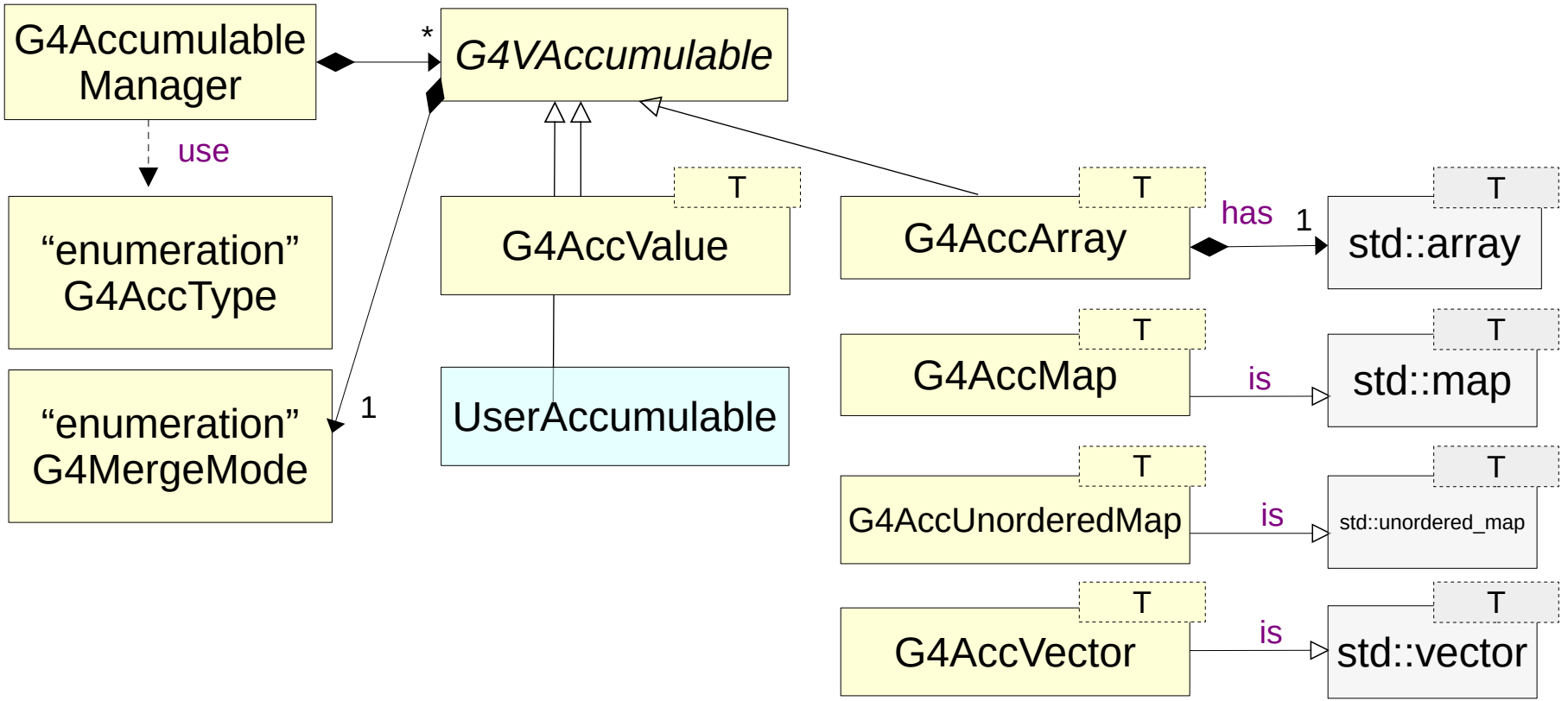
New developments in accumulables
For 11.3

First Accumulables Design

- The accumulables objects are named variables registered to the accumulable manager, which performs their merging in MT mode according to their defined merge mode.
- G4AccumulableManager is a (thread local) singleton
  - Has `std::map<G4String, G4VAccumulable*>`
- Provides functions to Register an accumulable
- And Merge() to merge of all registered accumulables
  - The merge mode can be selected per parameter
- Users can define their own accumulables
  - Tested with std::map<G4String, G4int> used for processes counting in TestEm* examples
- The first version of code introduced in 2015 used "Parameter" in classes names, changed in **"Accumulable"** next year



Accumulables Design in 11.3



G4AccType

```
// Enumeration for definition
// available accummulables

enum class G4AccType {
  kValue,          // G4AccValue<T>
  kArray,          // G4AccArray<T>
  kMap,            // G4AccMap<T>
  kUnorderedMap,   // G4AccUnorderedMap<T>
  kVector,         // G4AccVector<T>
  kUser            // User type
};
```

- Used by the accumulable manager to access the concrete accumulable type from the base type

G4VAccumulable

- The G4VAccumulable interface is extended with
- **Data members**: name, merge mode and id
  - Name and merge mode was previously defined only in G4Accumulable<T>
  - The id (added) can be used for fast access to accumulables or limiting eg. Print function to a selection of range of accumulables
- **Member Functions:**
  - `Set/Get` functions for new data
  - `GetType` – returns the new G4AccType enum value
    - `kUser` is returned by default (no need to update the user code)
    - It is overridden in all accumulables types implemented in Geant4
  - `Print(G4PrintOptions options = G4PrintOptions())`
    - The `'options'` parameter can be ommited, the output will then include 'name' and 'type'
    - All accumulable classes implemented in Geant4 provide the `Print` function implementation
    - User code migration is needed only if a function `Print()` with no or different arguments exists

G4Acc* Collections

- New classes that define collections of accumulables for most frequent collections of standard library: `array, vector, map, unordered_map`:
- G4AccArray, G4AccMap, G4AccUnorderedMap, G4AccVector
  - They simplify definition of collections of accumulables, which up to now have to be implemented by users
- The accumulable collections classes for **maps and vectors** are **derived** from the standard library collections
- The accumulable collection for **arrays**, as the `std::array` is not intended to be extended by inheritance, is implemented via **composition**

G4Acc* Collections - Constructors

- All collection classes provide sets of contructors with the same list of parameters as the std library collections + parameters specific to G4Acc: **name, merge mode**
- The constructors with brace initialization are supported for all collections
- **Name**, when provided, is always the first parameter
- **Merge mode** is always optional and it is always placed as the first optional parameter after the parameters without a default value, the default is G4MergeMode::kAddition

```
using MyArrayType = G4AccArray<G4double, 2>;
MyArrayType array1;
```

```
MyArrayType array2{0., 0.};
```

```
MyArrayType array3("array2");
MyArrayType array4{"EdepArray3", 0., 0.};
```

```
MyArrayType array5(
  "array2", 0., G4MergeMode::kMaximum);
```

G4AccValue<T>
G4AccumulableManager

- For consistency with new collections classes, G4Accumulable<T> is renamed into G4AccValue<T>
  - The old name is still available via the `using` directive for backward compatibility
- G4AccumulableManager functions
  - RegisterAccumulable and GetAccumulable renamed into Register() and GetAccValue() \
  - Overloading Register and new GetAcc* functions were added for all collection types, for example

```
template <typename T, std::size_t N>
G4AccArray<T, N>*
GetAccArray(const G4String& name, G4bool warn) const
```

  - Added Print(G4PrintOptions options = G4PrintOptions()) functions
  - The functions with the old names are deprecated
    - A compilation warning is issued when user compile the code with old names

Example of G4AccArray

```
// Construct (with default constructor)
G4AccArray<G4double, 2> myArray;

// Register
auto accManager = G4AccumulableManager::Instance();
accManager->Register(myArray);

// Fill/update
myArray[0] += edep;
myArray[1] += edep*edep;

// Merge (all registered accumulables)
accManager->Merge();

// Print
myArray.Print();
```

- Currently the accumulable collections are used only in the enhanced **test08**
- They can be demonstarted in some example(s) as a replacement of a user defined accumulable implementing a collection

Test08 Extensions

- Added data members of all new collections types, including various constructors (in the RunAction class)
  - The data of the same type are then saved also in vectors to simplify testing
- Added tests for accumulable acces by Name and by Id
- Added test for printing
  - All collections include the same 2 elements as the value data `fEdep` and `fEdep2` (accumulated edep and edep*edep values) what allows to check that merging gives the same results

Other Developments = Fixes

Fixes in Analysis Module

- Fixes in 11.2:
  - Fixed creating histograms with user defined bins. Addressing bug report #2541.
  - Fixed implementation of set commands per dimension: `"hn|pn/setX|Y|Z"`. Addressing a problem report in Geant4 forum.
  - Do not create setAxis* commands for idim = 4.
  - Fixed Coverity in G4RootHnFileManager
- Fixes since 11.2:
  - Fixed wrong conversion to G4String in G4THnToolsManager, gcc compilation error with C++23 Standard enabled (thanks to Ben Morgan)
  - Fixed compilation warning on macOS/XCode for implicit type conversion (thanks to Gabriele Cosmo)
  - Coverity fixes: use `std::move, const auto&` instead of `auto` to avoid copying

Fixes - 2

- Fixes in **g4tools** (Guy Barrand):
  - Current tools version **6.3.2.**
  - Coverity fixes:
    - use `std::move`, pass by reference to avoid copying
    - fixed a "divide by zero" medium issue in `axis::avoid_labels_overlap()`
- Ongoing fix addressing the bug report #2625
  - Fix setting file compression level in G4GenericFileManager: propagate setting to all registered file managers
  - Fix in applying the compression factor in `tools::wroot` is now under development & discussion in bugzilla

Plans

- 2024 Work plan items:
  - Accummulables: add support for most frequent std collections (array, vector) – (1)/(2) - DONE
  - Regular maintenance & extensions – (1)/(2) – ONGOING
- Beyond 2024
  - No new users requirements
  - Code maintenance