Norfolk, Virginia, USA • May 8-12, 2023
CHEP
2023
Computing in High Energy & Nuclear Physics

Analysis Tools in Geant4

I. Hřivnáčová, G. Barrand

IJCLab, Université Paris-Sud, CNRS-IN2P3

26th International Conference on Computing in High Energy and Nuclear Physics,
May 8 - 12 2023, Norfolk

Introduction

- Geant4 analysis category provides the users a “light” analysis tool
  - Available directly with Geant4 installation (without a need to link their Geant4 application with an external analysis package)
  - It allows to create and manipulate histograms and ntuples, and write them in the ROOT, AIDA XML, CSV and HDF5 file formats
- Based on the **g4tools** compact package
  - A light and easy to install set of C++ classes that can be used to perform analysis and visialization
  - Pure header code
    - Highly portable (including iOS, Android). Easily embeddable (no “config.h” or specific build tool in the way)
  - Strongly OO, no implicit management, thread safe (no writable statics)
  - See also: http://gbarrand.github.io
- Fully integrated in the Geant4 framework
  - Follows Geant4 coding style and also implements the built-in Geant4 user interface commands

Example of User Code

Extremely simple use:

- All operations are delegated to a **single manager** class
- Addressing to objects via integer indices
- After **booking** the objects, besides their **filling**, users have to call only few manager functions: `OpenFile()`, `Write()` and `CloseFile()`
- Hiding more complex operation behind these user calls (Eg. merging histograms on the `Write()` call)

```
#include "G4AnalysisManager.hh"

// Create analysis manager
auto analysisManager = G4AnalysisManager::Instance();
analysisManager->SetVerboseLevel(1);

// Book histograms, ntuple
analysisManager->CreateH1("Eabs","Edep in absorber", 100, 0., 800*MeV);
analysisManager->CreateH1("Egap","Edep in gap", 100, 0., 100*MeV);
//
analysisManager->CreateNtuple("B4", "Edep and TrackL");
analysisManager->CreateNtupleDColumn("Eabs");
analysisManager->CreateNtupleDColumn("Egap");
analysisManager->FinishNtuple();

// Open an output file
G4AnalysisManager::Instance()->OpenFile("B4.root");

// Fill histograms, ntuple
analysisManager->FillH1(0, value);
analysisManager->FillH1(1, value);
analysisManager->FillNtupleDColumn(0, eabs);
analysisManager->FillNtupleDColumn(1, egap);
AnalysisManager->AddNtupleRow();

// Save histograms & ntuple
analysisManager->Write();
analysisManager->CloseFile();
```

Code extracted from Geant4 basic example B4

Functionalities Overview

Basic Functionalities

- **First version (2011)**
- 1,2 – dimensional histograms
- Single ntuple with columns of int, float and double types
- Single file output
- Cvs, Root, Xml, HBOOK
    - HBOOK format was dropped in 2016

See also:

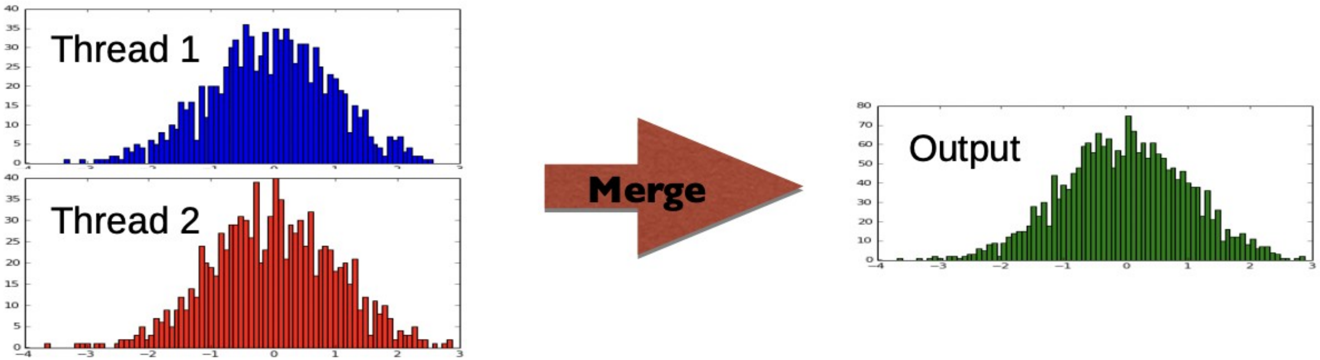
- *Integration of g4tools in Geant4 (CHEP 2013)*

- **Current (2023)**
- 1,2,3 - dimensional histograms, 1,2 dimensional profiles
- Ntuples with columns of int, float, double and string types and vectors of these types
    - Single ntuple limitation removed in 2013
- Multiple file output including multiple output formats
    - Single file limitation removed in 2021
- Csv, Root, Xml, HDF5
    - HDF5 added in 2018

Functionalities Overview

Multi-threading & MPI

- The analysis code has been adapted for multi-threading as well as all other Geant4 categories in Geant4 10.0 (2013)
  - Automatic merging histograms with MT (2013), with MPI (2015)
- Merging on flight of ntuples (MT, MPI) with the Root format (2017)
  - Merging options: column-wise (2016), row-wise(2017)



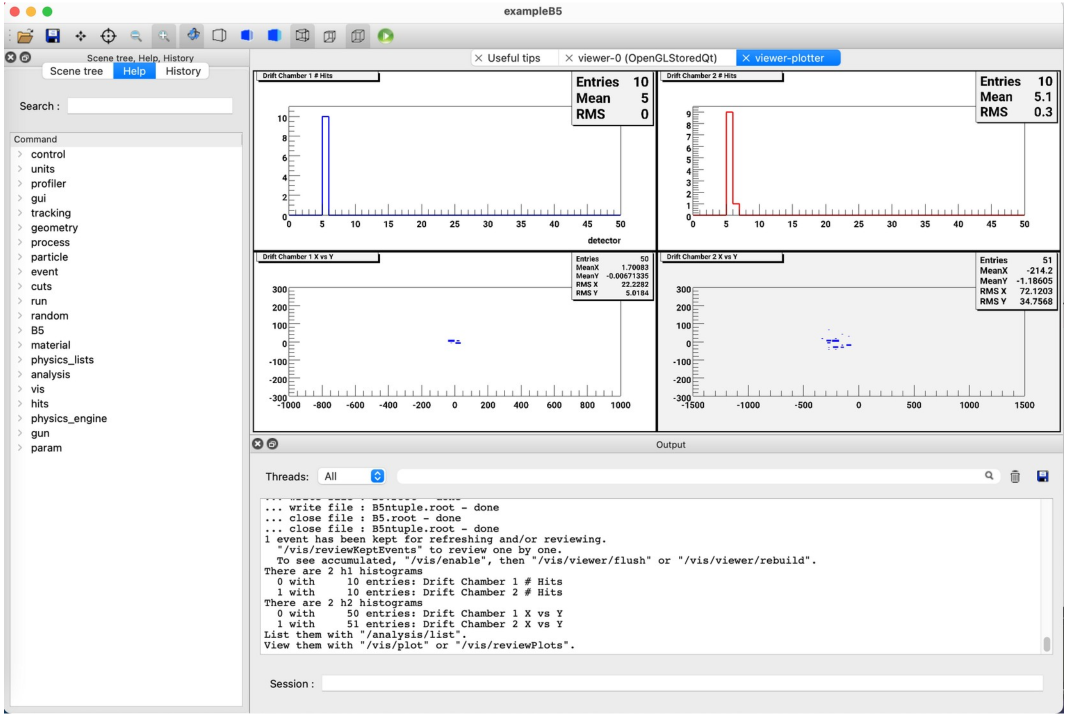
See also:

- *[Analysis Tools in Geant4 10.2 and 10.3 (CHEP 2016)]*
- *[HDF5 and row-wise ntuple in analysis tools in Geant4 10.4 (CHEP 2018)]*

New Functionalities Overview

Batch & Interactive Graphics

- Interactive plotting since 2021
  - The Geant4 visualization system is equipped to be able to do plotting, then to have a representation (a plot) of 1D or 2D histograms within a Geant4 visualization viewer.
- Currently only new **ToolsSG** visualization driver has this feature
- Batch plotting (since 2016)
  - Users can activate plotting for selected histograms or profiles, the plots will be then saved in a Postscript file

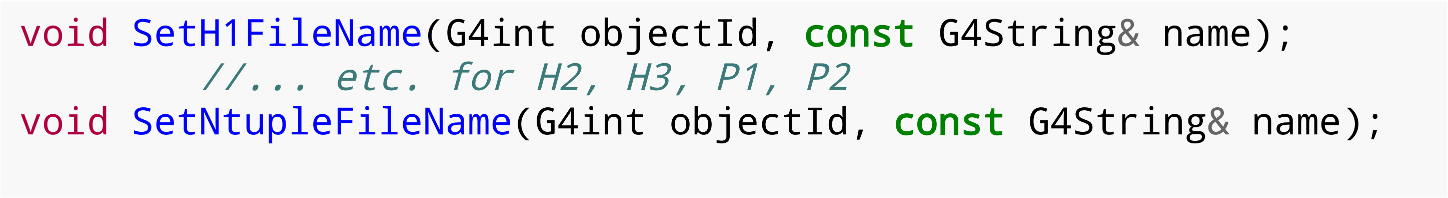


```
... write file : B5ntuple.root - done
... close file : B5.root - done
... close file : B5ntuple.root - done
1 event has been kept for refreshing and/or reviewing.
  "/vis/reviewKeptEvents" to review one by one.
  To see accumulated, "/vis/enable", then "/vis/viewer/flush" or "/vis/viewer/rebuild".
There are 2 h1 histograms
  0 with     10 entries: Drift Chamber 1 # Hits
  1 with     10 entries: Drift Chamber 2 # Hits
There are 2 h2 histograms
  0 with     50 entries: Drift Chamber 1 X vs Y
  1 with     51 entries: Drift Chamber 2 X vs Y
List them with "/analysis/list".
View them with "/vis/plot" or "/vis/reviewPlots".
```

New Functionalities Overview

Multiple Files Output

- Since 2021, users can choose to write selected objects in a different file than the default one using the G4AnalysisManager functions



```
void SetH1FileName(G4int objectId, const G4String& name);
        //... etc. for H2, H3, P1, P2
void SetNtupleFileName(G4int objectId, const G4String& name);
```

- It is possible to mix output types for histogram and profiles objects, only one output type is supported for ntuples.
- The file names should be provided with an extension (.csv, .hdf5, .root or .xml) unless a default file type is set with a dedicated function
- The corresponding UI commands are also available

New Functionalities Overview

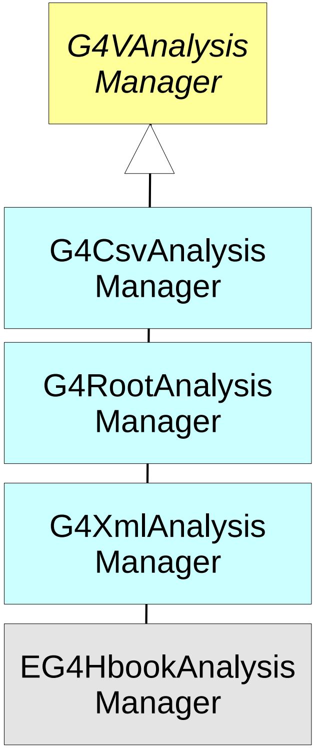
Object Cycles

- Writing the same histogram/profile/ntuple on file several times
- Automatically attributed the cycle number
  - Root IO supports the cycles naturally
    - When eg. a histogram “`myhisto`” is written more times, we can see in the browser `myhisto;1 myhisto;2` etc.
  - For the other output types, the cycle number is appended after the object name:
    - Eg. `myhisto.csv`, `myhisto_v2.csv`, etc.
- Since 2022

```
/run/initialize
/analysis/openFile B5.root
#
/run/beamOn 30
/analysis/write
/analysis/reset
#
/run/beamOn 30
/analysis/write
/analysis/reset
#
/analysis/closeFile
```

An example of a Geant4 run macro with 2 write cycles

Design Evolution
First Design (2011)

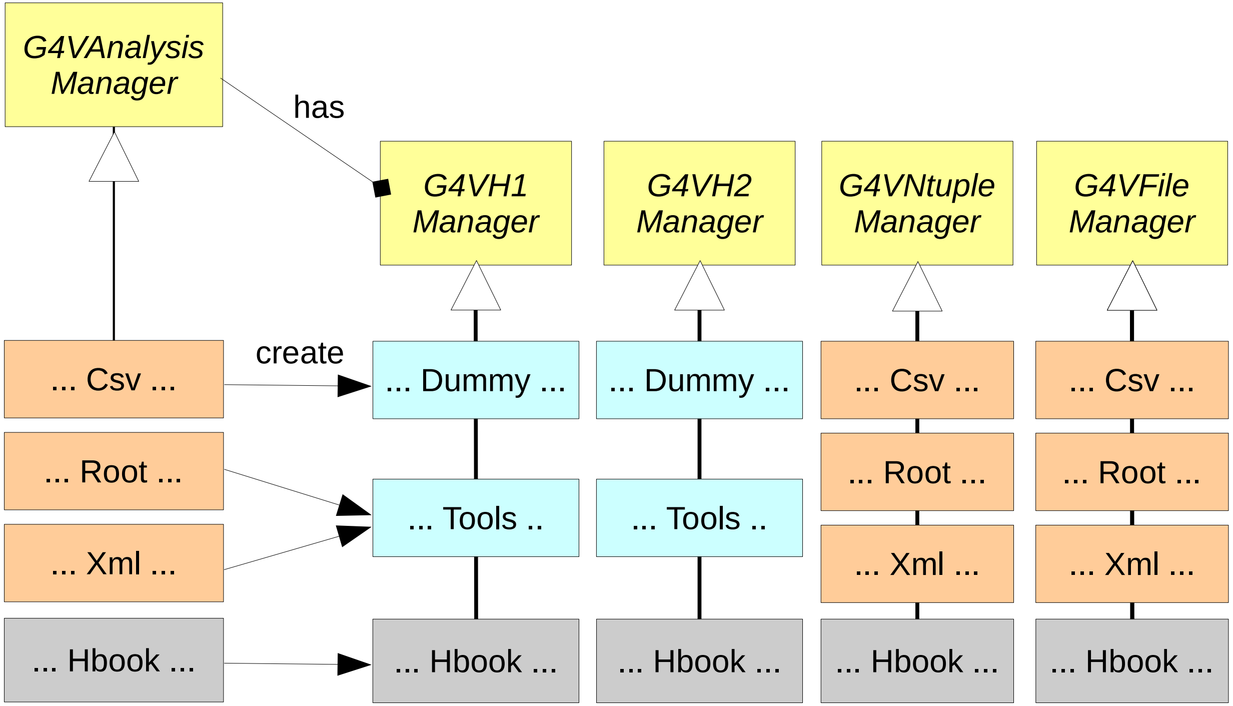


- Simple inheritance
- Common base class:
  - Interfaces functions non dependent on technology
- Manager classes:
  - Implement base class interfaces and output specific access functions
- All managers could be used in an identical way via a generic G4AnalysisManager type defined as `typedef` in dedicated header files:
  - `g4csv.hh`, `g4root.hh`, `g4xml.hh` and `g4hbook.hh`

- *HBOOK specific classes were provided in examples, as they required linking with CERNLIB*

Design Evolution

NVI Design Pattern (2013)



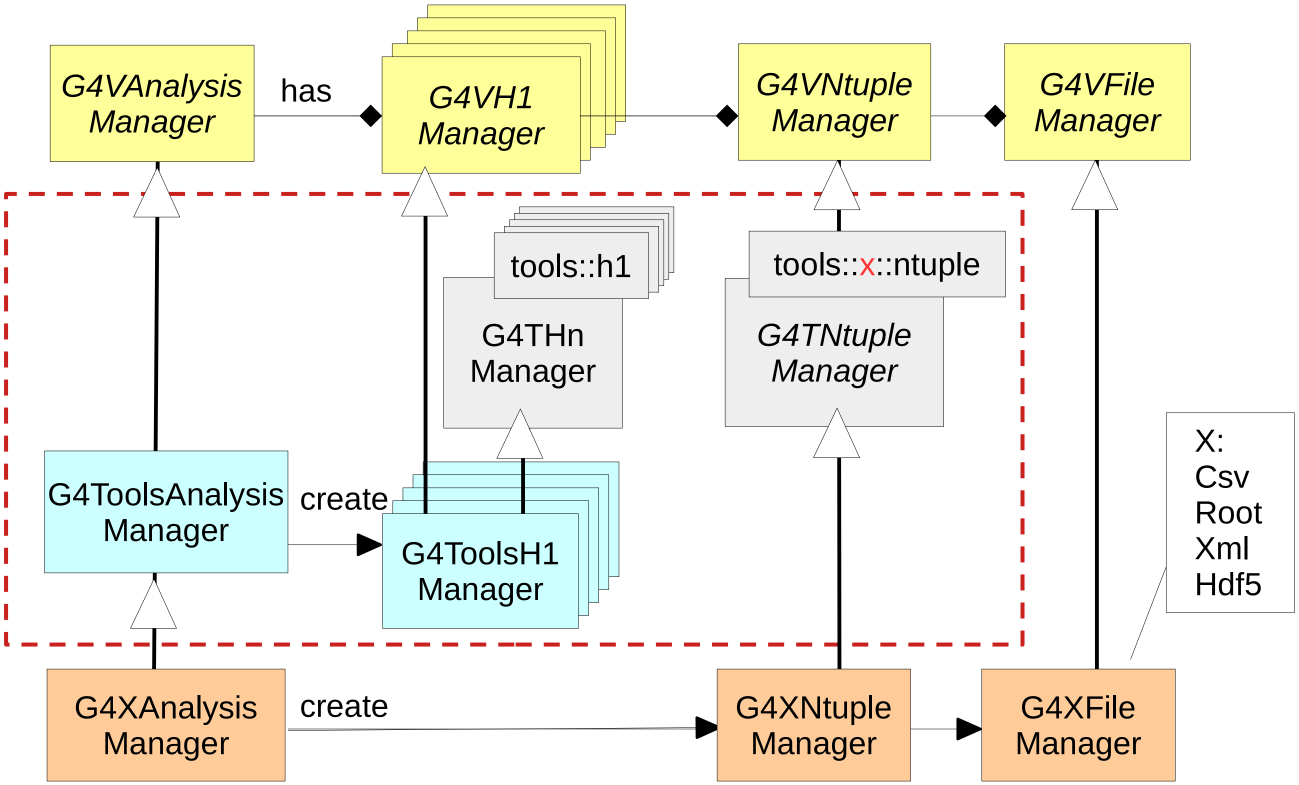
- NVI = Non Virtual Interface pattern
- Non virtual public methods, defined in G4VAnalysisManager, call protected, pure virtual members in the (friend) component classes, which are implemented in the output specific classes

- *CVS histograms were provided only since 2014*

See also [Integration of g4tools in Geant4 (CHEP 2013)](#)

Design Evolution

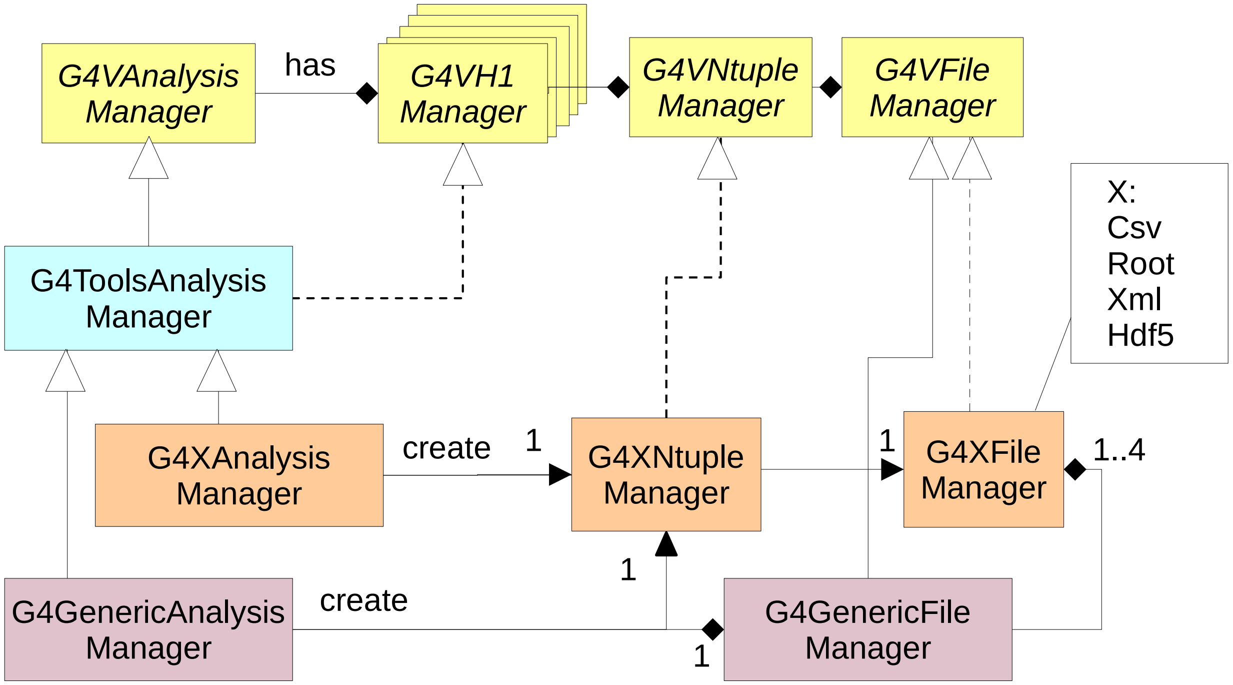
Tools Analysis Manager (2017)



- Added G4ToolsAnalysisManager class for common handling all histograms/profiles
- Templates were introduced where suitable to avoid code duplications in the previous Geant4 version
  - They concerned only analysis classes internals and did not affect the API seen by the users

Design Evolution

Generic Analysis Manager (2020)



- *The dashed lines are used for more complex implementations not displayed in the diagram in detail*

- G4GenericAnalysisManager takes the role of the top analysis manager class
- The file managers can be more than one, they are then handled by the generic file manager
  - Allows mixing output file formats for histograms and profiles
- Only one ntuple file manager
- Since 2021 - default G4AnalysisManager type defined via `using` in new header file `G4AnalysisManager.hh`

User Interface Stability

- The adopted design (non-virtual public interface, provided with the G4VAnalysisManager class) turned out to be of **an excellent choice in terms of the user code stability**
- Very few user code migration items [*]) over the past decade:
  - 10.3: **Stopped** support of the HBOOK output format
  - 10.4, 10.5, 10.6: Changes in the setting **option for ntuple merging** (added/modified arguments for merging mode)
  - 11.0 (major Geant4 release)
    - New G4AnalysisManager.hh **header file** replaced the output format specific headers “g4csv.hh”, “g4root.hh”, “g4xml.hh” and “g4hdf5.hh” and removed earlier introduced alternative factory methods
    - Migration to G4ThreadLocalSingleton in all analysis manager and reader classes. The singleton instances are now deleted by the Geant4 kernel; their **explicit deletion** in client code has to be removed

[*]) The items for user code migration are listed in the Geant4 release for every release

Code Quality, Modernization

- Geant4 Infrastructure
  - Continuous integration testing
  - Coverity Statistic Analysis Tool
  - Coding guidelines following new C++ standards (updates for C++11, C++17)
- Test codes – in geant4-dev/tests
  - Test03 – test all object types with most of options with all output specific and generic managers
  - Test08 – accummulables
  - Test32 – test multiple file outputs
- C++11 upgrades in 2015, C++17 since 2021
  - `auto`, `nullptr`, shared and unique pointers, `using`, deleted constructors, …
  - structured binding, `filesystem`, ...
- Clang-tidy checks (2021-2022) as recommended in the Geant4 coding guidelines
  - Checks from `performance-*`, `modernize-*` and `readability-*` families

Conclusions

- The Geant4 analysis tools are available for users since more than 10 years
- Started with a simple set of classes providing a limited set of functionalities, it has been continously enhanced with new features almost every year
- We gave an overview of the recent developments and discussed our design choice that allowed its evolution including major upgrades with minimum impact on the user code

- See more:
- Analysis section in the Geant4 Application Developers Guide
- Geant4 code on GitHub

Backup Slides

g4tools

- A light and easy to install set of C++ classes that can be used to perform analysis in a Geant4 program. It allows to create and manipulate histograms and ntuples, and write them in the ROOT, AIDA XML, CSV and HDF5 file formats
- Originally part of the inlib and exlib packages
  - The `inlib` and `exlib` namespaces automatically changed to `tools/toolx` in the g4tools (Geant4) distribution in order to avoid potential namespace clashes with other codes
- Pure header code
  - Highly portable (including iOS, Android). Easily embeddable (no “config.h” or specific build tool in the way)
- Strongly OO, no implicit management, thread safe (no writable statics)
- Included directly in Geant4
- See also: http://gbarrand.github.io

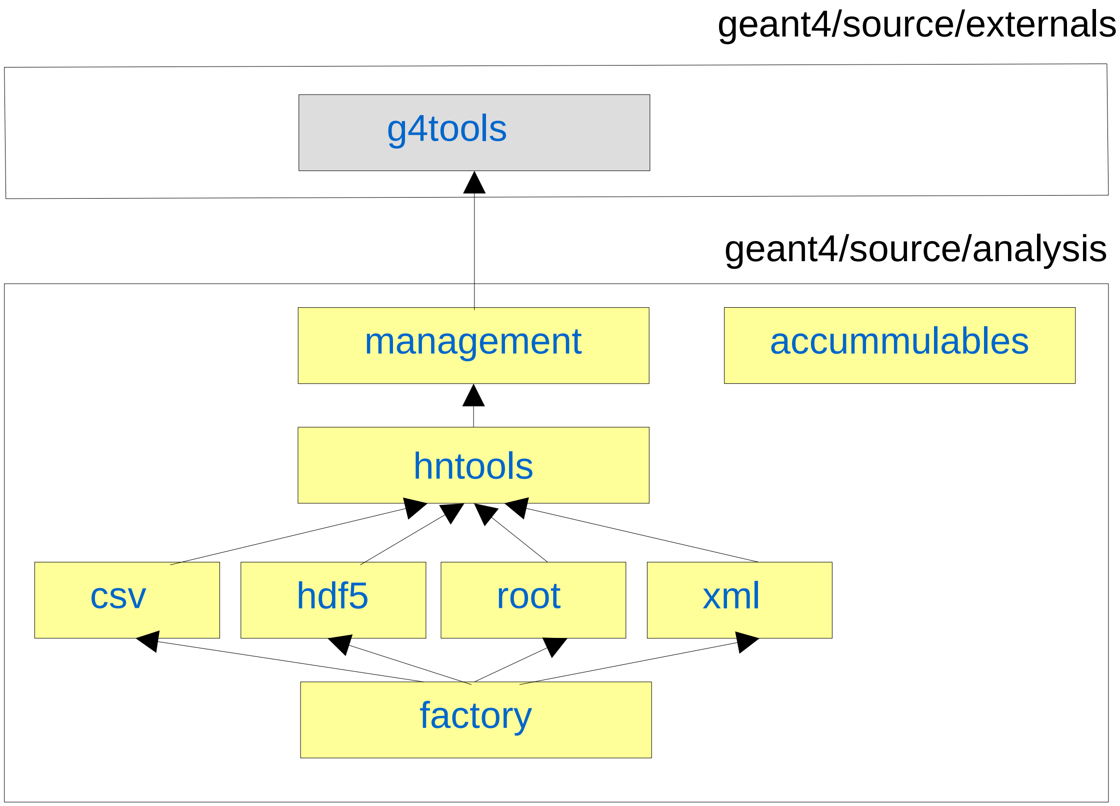
Functionalities Overview

Further Components

- **Reader Classes**
- Classes for reading back the files produced with the analysis managers
- All date types and file formats are supported
- Since 2014

- **Accummulables**
- Named variables registered to the accumulable manager, which provides the access to them by name and performs their merging in multi-threading mode according to their defined merge mode.
- Used in Geant4 examples
- No dependencies on g4tools and the analysis managers
- Since 2015

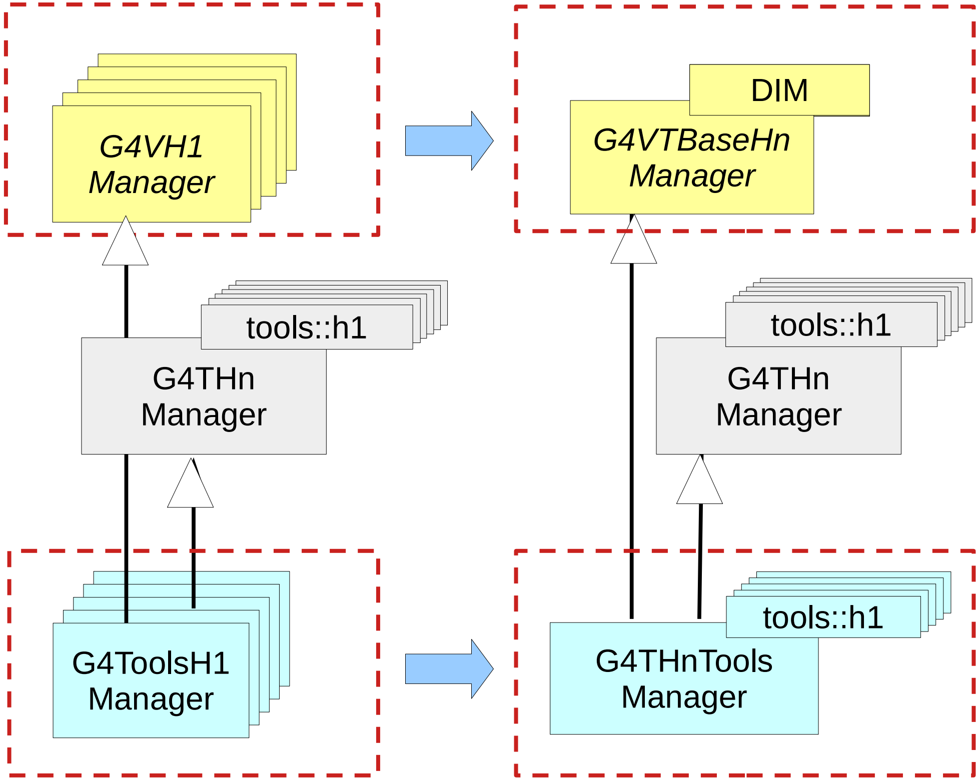
Code Organization



- G4tools - code to write and read histograms and ntuples in several formats
  - The output in HDF5 format (since Geant4 10.4) requires the HDF5 libraries installation
- Analysis category
  - Higher-level management of the g4tools objects (files, histograms and ntuples), handle allocation and removal of the objects in memory and provide the access methods to them via indexes.
  - Accummulables – a standalone set of classes not dependent on g4tools

Design Evolution

More Templates (2022)



Common implementation of **histogram handling** for all histogram/profiles type

- 15 classes were reduced to 3
- 1 interface: G4VTBaseHnManager<DIM>, 1 implementation: G4THnToolsManager<DIM,HT> and 1 messenger (not included in the picture)
- Instead of 5 interfaces, 5 implementations, 5 messengers for each histogram/profile type (H1, H2, H3, P1, P2)