Integration of g4tools in Geant4

Ivana Hřivnáčová
Institut de Physique Nucléaire (IPNO), Université Paris-Sud, CNRS-IN2P3, Orsay, France
E-mail: ivana@ipno.in2p3.fr

Abstract. gtools, that is originally part of the inlib and exlib packages, provides a very light and easy to install set of C++ classes that can be used to perform analysis in a Geant4 batch program. It allows to create and manipulate histograms and ntuples, and write them in supported file formats (ROOT, AIDA XML, CSV and HBOOK).

It is integrated in Geant4 through analysis manager classes, thus providing a uniform interface to the gtools objects and also hiding the differences between the classes for different supported output formats. Moreover, additional features, such as for example histogram activation or support for Geant4 units, are implemented in the analysis classes following users requests. A set of Geant4 user interface commands allows the user to create histograms and set their properties interactively or in Geant4 macros. g4tools was first introduced in the Geant4 9.5 release where its use was demonstrated in one basic example, and it is already used in a majority of the Geant4 examples within the Geant4 9.6 release.

In this paper, we will give an overview and the present status of the integration of g4tools in Geant4 and report on upcoming new features.

1. Introduction
Analysis tools based on AIDA (Abstract Interfaces for Data Analysis) [1] were used in Geant4 [2] examples since the Geant4 3.0 release (December 2000). No analysis code was provided in Geant4 source until 2010.

There are several AIDA compliant tools linked in the Geant4 Guide for Application Developers [3]: JAS[4], iAIDA[5], Open Scientist Lab[6], rAIDA[7]. However some of them are not kept maintained, they do not all implement the AIDA interfaces completely and also they are not always easy to be installed and used.

The new analysis category based on g4tools [8] was added in the Geant4 9.5 release with the aim to provide the users a “light” analysis tool available directly with Geant4 installation without a need to link their Geant4 application with an external analysis package.

2. Analysis Category in Geant4
The new analysis category in Geant4 consists of the analysis manager classes and it includes also the g4tools package. It is used in Geant4 examples since two years. Some interest from Geant4 users has been shown by posting questions and bug reports in the users forum. The complete migration of all Geant4 examples to its use is planned for the Geant4 10.0 release. Just one example with a pure AIDA-based analysis will be kept within a dedicated examples set (in extended/analysis category), together with a g4tools-based and ROOT-based analysis codes implemented for the same physics scenario.
2.1. \texttt{g4tools}

\texttt{g4tools} provides code to write histograms and ntuples in several formats: ROOT \[9\], XML AIDA \[1\] format, CSV (comma-separated values format) for ntuples and HBOOK \[10\]. It is a part of inlib and exlib \[8\] libraries, that include also other facilities like fitting and plotting.

\texttt{g4tools} is included directly in Geant4. In order to avoid potential namespace clashes with other codes using inlib/exlib, e.g. to do Geant4 visualization, the \texttt{inlib} and \texttt{exlib} namespaces are automatically changed to \texttt{tools} in the \texttt{g4tools} distribution in Geant4. The \texttt{g4tools} code is “pure header”, i.e. all code is defined as inline and no library is built from it.

2.2. Analysis Classes

The analysis classes provide a uniform, user-friendly interface to \texttt{g4tools} and hide the differences according to a selected output technology from the user.

They take care of a higher level management of the \texttt{g4tools} objects (files, histograms and ntuples), handle allocation and removal of the objects in memory and provide the access methods to them via indexes. The possibility to activate or inactivate selected histograms was also implemented on users request.

The analysis classes are fully integrated in the Geant4 framework: they follow Geant4 coding style and also implement the built-in Geant4 user interface commands that can be used by users to define or configure their analysis objects. The CLHEP \[11\] system of units, used in Geant4, is also integrated. Users can set a selected unit to the histogram values, then filled values are automatically converted to the selected unit and the unit also appears in the histogram axis title.

For a simplicity of use all user interface analysis functions are provided within a single class which is seen by the user as \texttt{G4AnalysisManager}. Internally, this type is defined via a \texttt{typedef} and it can point to one of four output type specific manager classes:

- \texttt{G4CsvAnalysisManager}
- \texttt{G4RootAnalysisManager}
- \texttt{G4XmlAnalysisManager}
- \texttt{ExG4HbookAnalysisManager}

All these manager classes implement the common interfaces, specific access functions (with a return type specific to the output format) and the singleton access method \texttt{Instance()}.

In order to avoid a dependence of the Geant4 kernel libraries on CERNLIB \[12\] the \texttt{ExG4HbookAnalysisManager} class is not included in the analysis class category in Geant4 source but in examples (\texttt{examples/extended/common/analysis} directory) together with all necessary configuration files for its build with the CERNLIB libraries.

3. Class Design

In the first design, all analysis functions were implemented within each specialized manager class or its base class, see Fig. 1. While such a single-class interface is extremely easy to be used in a client code, it is difficult to be maintained and extended in a long term. It also makes difficult code sharing between more manager classes.

In new design, see Fig.2, the monolithic manager classes are split into smaller classes per analysis object type: one-dimensional and two-dimensional histograms (\texttt{h1, h2}), ntuple, file. Beside considerable reducing of the class sizes this also allows sharing implementation. The \texttt{Root} and \texttt{Xml} managers use the same \texttt{h1} and \texttt{h2} \texttt{g4tools} objects and so the same \texttt{H1Tools} and \texttt{H2Tools} manager classes. The new design will also facilitate future extensions which are foreseen for one-dimensional and two-dimensional profiles to support users request.

The top manager class providing the complete access to all interface functions is kept. However its interface is provided in terms of non virtual public functions (following the so-called
Non Virtual Interface pattern [13]) which are implemented via calls to pure virtual members in the friend component classes, implemented in technology specific classes. This makes the public interface robust and less sensitive to internal implementation changes.

![Figure 1. The first class design.](image1.png)

![Figure 2. New class design.](image2.png)

4. New Features

Besides the migration to multi-threading mode two new important extensions were added since the last Geant4 9.6 release on Geant4 developers requests.

4.1. Interface Extensions

A limitation for handling only one ntuple has been removed. The interface has been extended with new functions for handling more ntuples with keeping a backward compatibility of a user code. The new feature is demonstrated in Geant4 extended AnaEx01 example where the ntuple is split in two for this purpose.

The second extension is the support for a user defined binning scheme (already available in g4tools) and a selection of logarithmic binning scheme. The former can be defined using new `CreateH1`/`H2`() or `SetH1`/`H2`() methods where a vector of binning edges should be specified instead of a number of bins with minimal and maximal values. The logarithmic binning scheme can be selected using a new optional string argument (binSchemeName) added in already existing `CreateH1`/`H2`() or `SetH1`/`H2`() functions. When "log" value is selected (while "linear" is default), the edges for logarithmic binning are automatically computed. This selection is also available via Geant4 built-in user interface `/analysis/h1[2]/set` and `create` commands.

The next extension, support for one-dimensional and two-dimensional profiles is in development.

4.2. Multi-threading

Large effort is actually devoted by the Geant4 collaboration to support multi-threading mode in the Geant4 10.0 release. This is planned to be the first public release with both sequential and multi-threading build modes from the same source code. This release will be an accomplishment of the several-years-spanned work [14] of the Geant4 multi-threading team now followed by the whole Geant4 collaboration [15].

As well as all other Geant4 categories, the analysis code had to be adapted for multi-threading. The multi-threading specific code is completely hidden from the users. In multi-threading mode, the analysis manager instances are internally created on the master and thread workers and data accounting is processed in parallel on workers threads.
Histograms produced on thread workers are automatically merged on `Write()` call and the result is written in a master file. Merging is protected by a mutex locking, using `G4AutoLock` utility. Ntuples produced on thread workers are written on separate files, which names are generated automatically from a base file name, a thread identifier and eventually also an ntuple name. No merging of ntuples is performed.

The `HBOOK` output is not supported in multi-threading mode.

No changes are required in the user client analysis code for migration to multi-threading. `g4tools` classes have proved to be thread safe. No issues with concurrent processing were observed in Geant4 tests.

5. Conclusions
The analysis category has been included in Geant4 since two years. In this paper we gave an overview of its present status and reported on upcoming new features: a newly applied modular design, extended functionality and migration to multi-threading. The agreement on a migration to this new analysis tools in all examples in this year, as well as the activity in the analysis category of Geant4 user forum, show a growing interest of both Geant4 developers and users.

Acknowledgments
The author would like to acknowledge Guy Barrand from `LAL` for his help with `g4tools` integration as well as his responsiveness and flexibility in applying new requirements in `g4tools` code.

References
[1] http://aida.freehep.org
[2] Agostinelli S et al 2003 `Nucl. Instrum, and Methods` A506 250-303
  Allison J et al 2006 `IEEE Transactions on Nuclear Science` 53 No. 1 270-278
[3] http://cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/index.html
[4] http://jas.freehep.org/documentation.htm
[5] http://aida.dynalias.net/
[6] http://openscientist.lal.in2p3.fr/
[7] http://filessoft.desy.de/portal/software_packages/raida/
[8] http://inexlib.lal.in2p3.fr
  See also: Barrand G 2013 `Computing in High Energy and Nuclear Physics`: softinex, inlib, exlib, ioda, g4view, g4exa, wall
[9] http://root.cern.ch
[10] http://wwwasdoc.web.cern.ch/wwwasdoc/hbook_html3/hboomain.html
[11] Lönnblad L 1994 `Comput. Phys. Commun.` 84 307-316
  See also: http://cern.ch/clhep
[12] http://cernlib.web.cern.ch/cernlib
[13] Sutter H, Alexandrescu A `C++ Coding Standards: 101 Rules, Guidelines, and Best Practices` (Boston, MA: Addison-Wesley) 2005; p.68
[14] Dong X et al. 2010 `Computer Science` Vol 6272 287-303
  Dong X et al 2012 `J. Phys.: Conf. Ser.` 396 052029
[15] Cosmo G 2013 `Computing in High Energy and Nuclear Physics`: Geant4 - Towards major release 10