

Geant4 Hadron Validation Framework Design Proposal

S. Banerjee, D. Elvira, H. Wenzel, J. Yarba
Geant4 Hadronic Group
Fermilab, Batavia, IL 60510, U.S.A.

Contents

1	Goal of this document	1
2	Introduction	2
3	Goals of the proposed Validation framework	2
4	Steps of the validation process	3
4.1	Executing Tests	3
4.2	Merging and Comparing results	5
4.3	Storing Results	7
4.4	Publishing	8
5	Prototype application	9

1 Goal of this document

This software design document (SDD) provides a written description of the proposed Geant4 Validation Framework. The intention of the document is to give the Geant4 collaboration and ultimately the software development team an overall guidance of the architecture and all components while maintaining a high-level view of the project. Recommended technology choices are given. The document specifying the requirements can be found in [1]. To make sure that the design choices are feasible and to gain experience we plan to provide a prototype to serve as proof of principle.

2 Introduction

Geant4 [2] is a software toolkit for the simulation of the passage of particles through matter. It is used by a large number of experiments over many application domains, high energy and nuclear physics, astrophysics and space science, medical physics and radiation protection. Geant4 describes the physics of all different particles over a wide energy region: from optical photon and thermal neutron to particles with energies of several TeV.

Geant4 divides the physics process into several categories: electromagnetic, hadronic, weak interactions, and optical. The electromagnetic physics processes include ionization, bremsstrahlung, multiple scattering, Compton and Raleigh scattering, pair production, photo-electric effect among others. All these processes are theoretically well understood and the descriptions match well with the existing measurements. The physics of hadrons and nuclei is not as well-understood from first principles and thus must be described in terms of one or more models. The main challenge is to provide a good description of the physics over a range of energy spanning 15 orders of magnitude. Several models are required to do this, even though a typical user may use only a few. These models are either driven by data, motivated by theory, or based on parametrization and extrapolation of cross sections. Each of these models is valid for a restricted number of incident particles and over a restricted energy region. It is essential to find out the range of application of these models by examining them against available data. The models are periodically improved by injecting new ideas and incorporating new experimental data.

3 Goals of the proposed Validation framework

Validation of a physics model is an integral part of commissioning the model in the Geant4 application and has been a part of Geant4 activity from the very early days. This work has been done within the Geant4 collaboration using published data and also by users with complete detector setups. This tests validating various aspects of the Geant4 framework are collected in the Geant4 CVS repository in the tests module. This tests are maintained and run by the various Geant4 developers. The goal of this proposed validation framework is to integrate the existing tests, to formalize the validation process, to make the test results available in a comprehensive and convenient way and document the history (e.g. to allow to track how a model improves in time and what is the effect of each change).

A test is a combination of one test executable and one starting condition. Each test shall have a name, creation date (CVS tag), the name of the test executable, a copy of the start condition (Geant4 macro), owners associated with it and a description providing what is being tested and what the results are expected to look like (see TABLE TESTDES in Appendix A).

This design document covers two types of tests for hadronic model validation:

- Regression Tests, which are comparison of physics output from release to release.
- Validation Tests, which are comparison of physics output with published data.

Tests such as unit tests and system tests are expected to be performed outside of this framework [3].

The metric of the comparisons will be developed over time. To the first approximation MC/data for validation and Chi-squared for regression will be used. A number of comparison plots with superposition of symbols specifying data point and histograms representing model predictions will also be available.

4 Steps of the validation process

The Validation framework performs the following actions:

- Executing the test.
- Merging and Comparing results with reference.
- Storing the results.
- Publishing

In the following chapters we describe these actions and framework components in more detail and list the technical choices for implementation.

4.1 Executing Tests

Geant4 currently has about 2 major releases a year and each of these releases typically has 1-2 patches per release cycle. In addition there are about 10 internal releases per year. For each release the hadronic validation requires to run in order of a few thousand tests requiring an estimated CPU usage of about a few thousand hours. To complete this within a week one would have to have a few hundreds slots available [1]. For the execution of the Geant4 Hadronic Validation tests we propose to rely on dedicated resources where sufficient number of predefined grid slots will be allocated to the project at CERN and at Fermilab ¹. Opportunistic use of resources on-site should also be possible so that one isn't restricted to the allocated slots but can use additional resources on the site when available. Other grid sites associated with participating institutions can be included, if interest is expressed. We think that using generic grid resources will require a big overhead that might not be justified by the scope of the validation project.

For the scope of the Geant4 Hadronic Validation efforts, the proposed approach will secure sufficient resources and will also present the following advantages:

¹This was the approach of the ILC detector concepts, to generate data for their Letter of Intent

- guaranteed run time environment.
- reference and current builds of Geant4 are provided via central install being exported to all worker nodes (e.g. /grid/app on the general purpose fermigrid).
- no matchmaker is necessary (no pilot job will be necessary).
- job monitoring and diagnostic is easier. It's easy to communicate with on-site grid operators.

Launching validation jobs will be done, in a transparent manner, from CERN or from Fermilab. Launching jobs will also be possible from a participating external portals other than CERN or Fermilab. However, installation of certain software maybe necessary (an example is the Virtual Data Toolkit (VDT) and a host certificate for the gateway machine).

- Authentication/authorization will be provided via an appropriate grid certificate. The certificate can be requested at the following site [4].
- The Geant4 Virtual Organization (VO) already exists. Registration with the Geant4 VOMS can be done via this site [5].
- At present, the Geant4 VO is not given high priority status at Fermigrid (see [6]). A dedicated (single) slot for the Geant4 VO exists at Fermilab, and can be used as a test bench.
- A tutorial how to use grid resources at Fermilab can be found at [7].

Required resources must include sufficient temporary storage as well as longtime storage. At present, temporary storage is provided in the central area (/grid/data on FermiGrid).

The following requirements will apply to the jobs and validation application (test):

- job execution/CPU time should not exceed 8 hours.
- jobs that take longer than 8 hours of CPU must be split into shorter, parallel jobs. Splitting can be done by e.g. different models, materials, energies, incident particles or other parameters of the test. Another option is to run parallel processes were each process generates a fraction of the total events necessary to complete the test.
- during job execution a local disk on a worker node will be used.
- the output of a job must be small enough such that results can be tar-ed up and sent to a temporary storage at job completion.

- output may be composed in a form of Root file containing histograms (mainly for physics results) and/or in a form of ASCII files, for log file and end-of-job report.
- end-of-job report must be composed at job termination. The contents of the end of job information is detailed in section 4.3 and a possible data base schema to store this information is listed in Appendix A.
- the output must be parse-able (xml?) such that retrieval of results can be automated.
- it must be possible to set a unique random seed for each job/process, to allow job splitting if necessary, to avoid production of identical samples and for the reproducibility of results.
- output of split jobs should be merge-able; however, merging of incompatible outputs must be prevented (for example, different starting conditions, etc.).
- mechanism to provide simple yet sufficient provenance, to ensure proper merging of fractional outputs (an example provenance can be naming convention for histograms and/or files)
- provide a well documented exit code indicating success or the reason for failure.

The proposed work-flow is schematically presented in Figure 1.

4.2 Merging and Comparing results

This step is executed after the execution of all processes/jobs finished. At this point

- the output of all the jobs is merged.
- create the plots showing the results with the reference superimposed.
- automatically run a comparison between the reference store and the test result or have the developer responsible for the specific test inspect the result by eye. The result of the test should get a score (e.g. passed, failed, improved...) and the developer can add comments (e.g. why the test got scored that way.)
- The developer decides if the test result gets published or discarded. In case it gets published the access level will be set (e.g. public, restricted to Geant4 collaboration..)

For this step two applications need to be developed. The application that merges the output and does the comparison. An application that allows the developer responsible for the test to submit the result to the data base including:

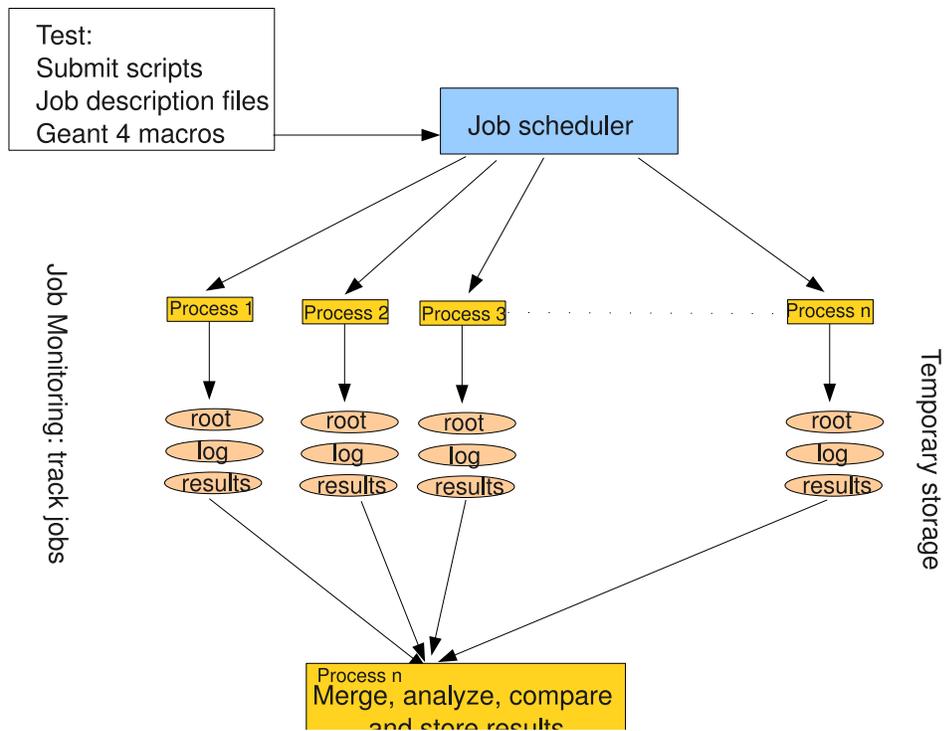


Figure 1: *Workflow of a Test.*

- either location of the plots/results so that the location can be referenced in the data base or the results and plots themselves are submitted as a blob.
- the test score
- comments
- the access level (e.g. public, restricted to Geant4 collaboration)

The application updating the database has the appropriate access privileges to the database. It needs to provide proper authentication and access control so that e.g. the developer for a given test can publish the results of the test he/she is responsible for. Like for the DisplayBrowser (see Section 4.4) a JSP based Web Application is our choice.

4.3 Storing Results

Data storage: way to store the validation data in the most convenient way. A relational database seems the most natural choice (PostgreSQL). Histograms can be either stored and retrieved as blobs or they can be stored in the file-system with a reference to the files stored in the database. The stored comparisons and histograms can then be accessed via a web application. Database and Web-application server can be deployed on one server with the option of separating the functionality on separate machines. The operations should be done by Geant4 designated administrator. The database needs to be set up in a secure way e.g. the database user associated with the Display application should only have the minimal necessary privileges.

Two kinds of storage will be provided: the reference store for long term storage in the order of 10 years and short term storage intended to store intermittent data for about 2 weeks.

For each job the following information should be stored after the output of all processes has been merged and the result has been evaluated.

- Site information
- unique ID identifying the JOB (as e.g. provided by Condor)
- total job CPU usage
- process CPU information
- process memory usage
- process local disk usage
- Operating system information

- Total CPU Usage
- time process was submitted
- time process started running
- time process finished
- Geant4 version
- name of the test
- description of the test and comments regarding the test e.g. what is being tested, Statistics needed etc.
- name of the executable (script that controls and executes the test)
- test score.
- type of comparison performed e.g. χ^2 , Kolmogorov Smirnov comparison with reference histogram.
- ... what else ?...

4.4 Publishing

Display Browser: this is the process making the data available on the web via a JSP (Java Server pages) web-application running on a Tomcat web application server. Java EE provides the framework to develop server side based web application. The development is done using the Netbeans IDE using Ant to build the application. The element of the proposed storage and display solution are shown in Figure 2.

- **Database connection:** the DisplayBrowser connects to the database via a connection pool. This is started at the start of the web application. Using a connection pool avoids that a new connection is established whenever a new database request is issued. Establishing a new connection is a fairly costly process in terms of database resources and should be avoided.
- **Security:** password file (needed to connect to the data base) is kept in a central configuration file and proper file protection ensures that only the web application account can access the file. To access the file we use the standard Java Properties class. The database will be set up so that only minimal necessary access is granted to the web application. We further will protect access to pages that are only intended for Geant4 collaborators. The simplest way to authenticate the user is probably a password (.htaccess file). The interface design will follow proper security guidelines. This include

- always validate user input.
- never construct any commands, file names, script names, or their arguments, based on user input.
- restrict allowed values returned from user as much as possible by:
 - * Selection lists or ranges.
 - * variable typing by string conversion.
 - * URL character encoding to catch specials
 - * Restrict strings to alphanumeric
 - * Pass values indirectly
 - * SQL bind variables
- **Deployment:** the web application will be deployed to the web application server via a war file (war: web application archive).
- **Documentation:** JavaDoc to document the classes for the developers.
- **Revision Control:** CVS, which is supported by the Netbeans IDE.
- **Look and Feel:** to define the look and feel of the application we will use cascading style sheets. That will allow us to change the look easily without having to rewrite a lot of html code. We will explore JavaServerFaces technology as a standard for building server side user interfaces. Figure 3 shows how the page could look like.

5 Prototype application

We chose to use test47 as an example to exercise the execution step and to gain experience preparing a Geant4 test job to run on FermiGrid. To execute the tests (e.g. on the general purpose FermiGrid) the following steps were performed:

- install the Geant4 reference installation and make it accessible from the worker nodes (On FermiGrid an nfs mounted area is visible on all the worker nodes. The Geant4 installation resides in /grid/app/geant4-validation)
- install root reference installation and make it accessible from the worker nodes.
- install the test program (test47).
- provide scripts to properly set up the run-time environment in a grid environment and to execute the test program. Provide the jdl file to submit the job to the grid. The scripts to run the test in a grid environment and jdl file to run on fermi grid can be found in the Geant4 CVS repository. This prototype scripts are available in the Geant4 CVS repository under geant4/tests/test47/grid. This scripts can be run interactively or in a grid environment.

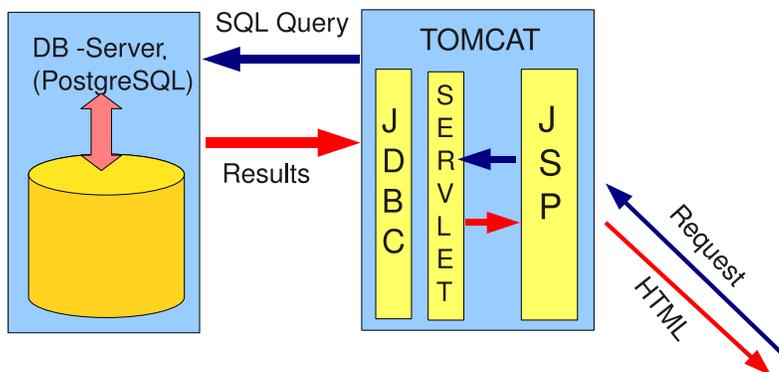


Figure 2: *Elements of Display Browser and Storage solution.*

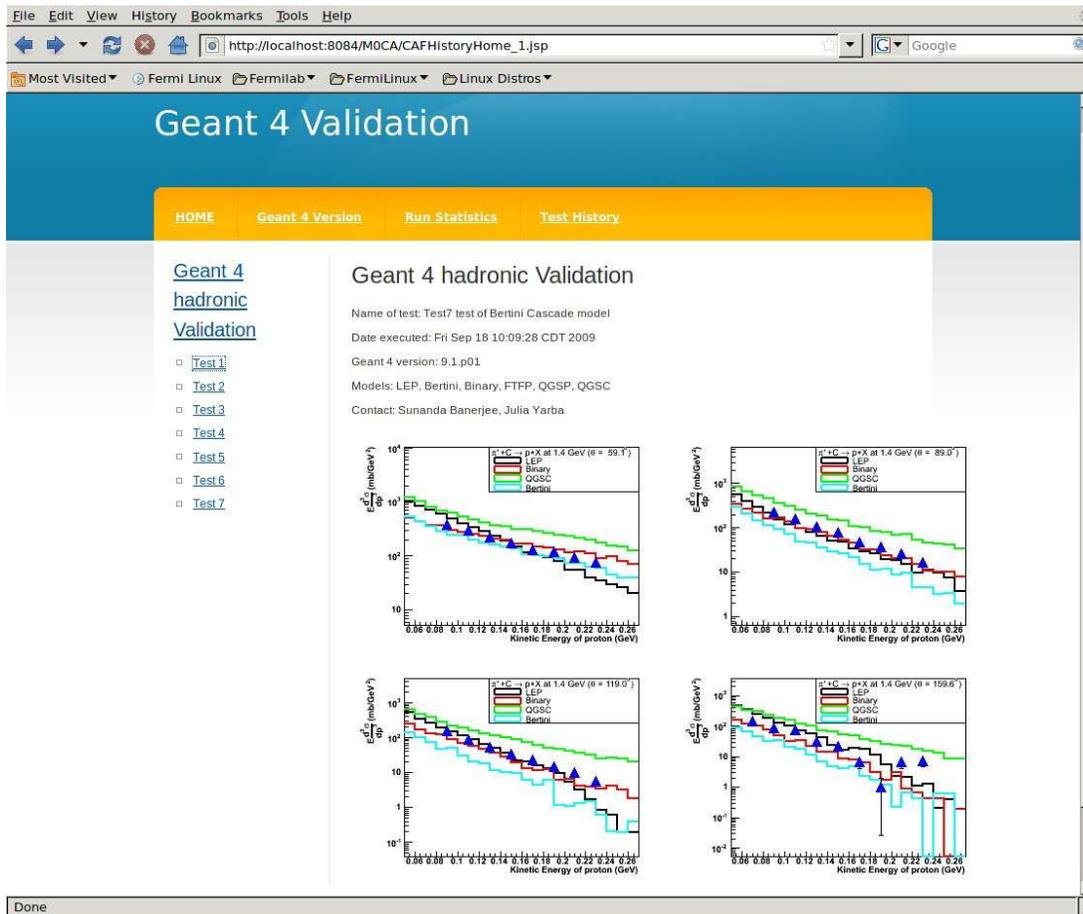


Figure 3: Possible 'Look and feel' of DisplayBrowser.

- collect, store the data and do the necessary bookkeeping. Evaluate that all processes finished successfully. If that's not the case take action like resubmitting the jobs that failed or if there is an identifiable problem fix that first and rerun the process or tag the test as failed.

Appendix A: Database schema

Below we show a draft of a possible data base schema. This schema is incomplete and will evolve in time as we finalize the design.

```
/*Table TEST_SCORE provides a dictionary listing the allowed test scores */
CREATE TABLE TEST_SCORE(
code  varchar(50),
Primary Key (code)
);
INSERT INTO TEST_SCORE VALUES('passed');
INSERT INTO TEST_SCORE VALUES('failed');
INSERT INTO TEST_SCORE VALUES('improved');

/* Table TESTTYPE provides a dictionary listing the defined test types */
CREATE TABLE TESTTYPE(
code  varchar(50),
Primary Key (code)
);
INSERT INTO TESTTYPE VALUES('chi2');
INSERT INTO TESTTYPE VALUES('byeye');
INSERT INTO TESTTYPE VALUES('KS');

/* Table TESTDES provides a description of each test */
CREATE TABLE TESTDES (
nameoftest      varchar(50),
nameofresp     varchar(50),
CVSTAG         varchar(50),
EXECUTABLE     varchar(50),
G4MACRO        varchar(50),
description     text,
UNIQUE (nameoftest,CVSTAG)
);

/* Table SITE provides a description of the Grid site where the test was run */
/* could include more info like : webpage of site, contact person... */
CREATE TABLE SITE (
nameofsite     varchar(50),
description    text,
UNIQUE (nameofsite)
);
```

```

CREATE TABLE TESTJOB(
SITE          varchar(50) references SITE(nameofsite),
Jobid         bigint,
CPUUsage     bigint
CPUinfo      varchar(50)
MEMusage     bigint,
DISKusage    bigint,
OSinfo       varchar(50),
TotCPUUsage  bigint,
submit_time  timestamp,
start_time   timestamp,
end_time     timestamp,
G4version    varchar(50),
testname     varchar(50) references TESTDES(nameoftest),
testexe      varchar(150),
testscore    varchar(50) references TEST_SCORE(code),
testtype     varchar(50) references TESTTYPE(code),
comment      text;
... what else ?...
UNIQUE (Jobid)
);

```

References

- [1] Geant4 Hadron Validation Framework Requirements
<http://cd-docdb.fnal.gov/cgi-bin/ShowDocument?docid=3165>
- [2] The two main reference papers for Geant4 are published in Nuclear Instruments and Methods in Physics Research A 506 (2003) 250-303, and IEEE Transactions on Nuclear Science 53 No. 1 (2006) 270-278.
<http://www.Geant4.org/Geant4/>
- [3] http://www.Geant4.org/Geant4/collaboration/working_groups/testing/index.shtml
- [4] DOE certificates can be requested at the following site:
<http://security.fnal.gov/pki/Get-Personal-DOEGrids-Cert.html>
- [5] Certificate can be properly registered with the Geant4 VOMS via this site:
<https://lcg-voms.cern.ch:8443/vo/Geant4/voms>
- [6] http://fermigrd.fnal.gov/grid_users/fermigrd_gpgrid.pdf
- [7] <http://confluence.slac.stanford.edu/display/ilc/How+do+I+use+the+OSG+Grid>