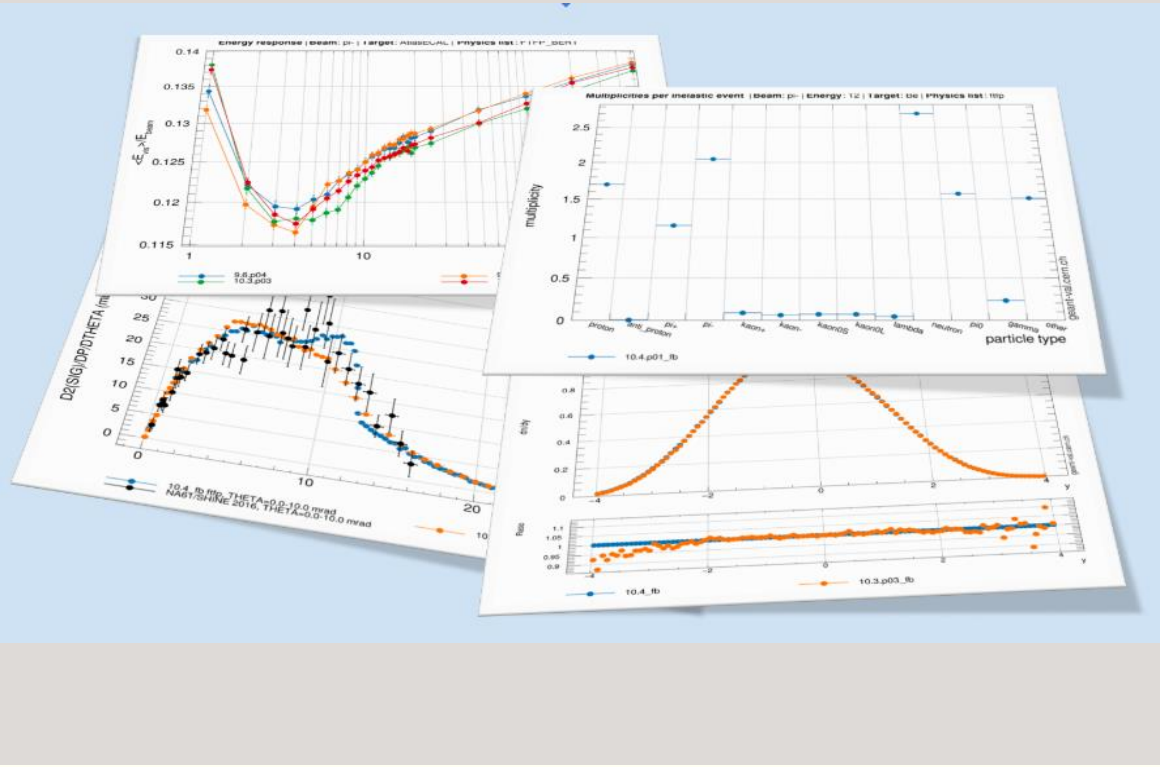
GEANT-VAL:
VALIDATION WEB APPLICATION
(+ USEFUL SIDE PRODUCTS)

Luc Freyermuth	EISTI, France
Dmitri Konstantinov	IHEP, Protvino
Grigory Latyshev	IHEP, Protvino
Witold Pokorski	CERN
Ivan Razumov	IHEP/CERN
Alberto Ribon	CERN

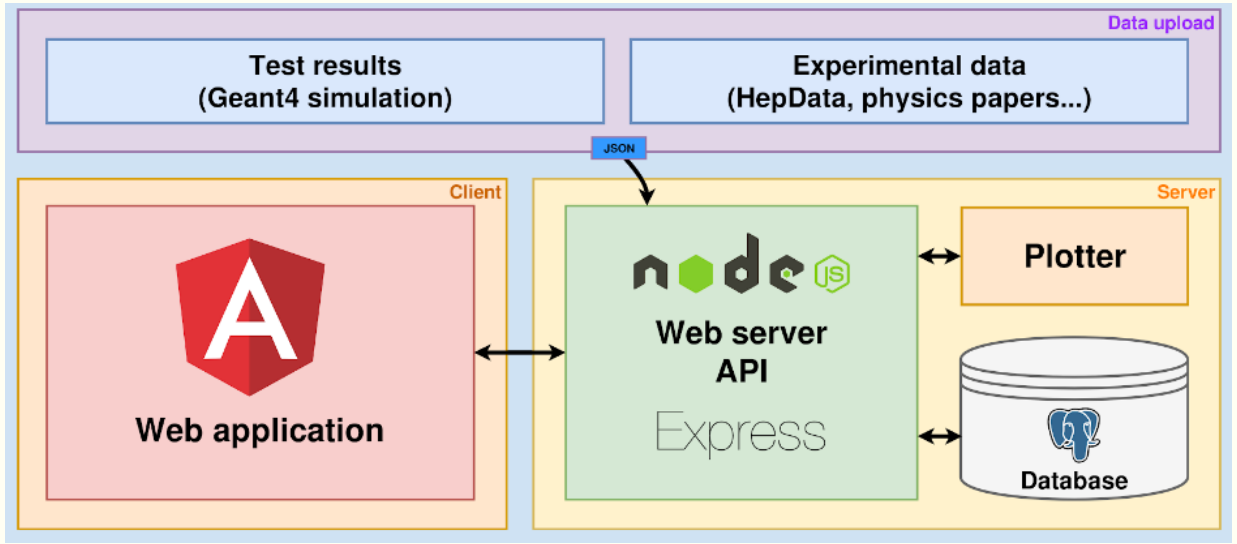
23rd Geant4 Collaboration Meeting, Lund, Sweden, 2018



Introduction

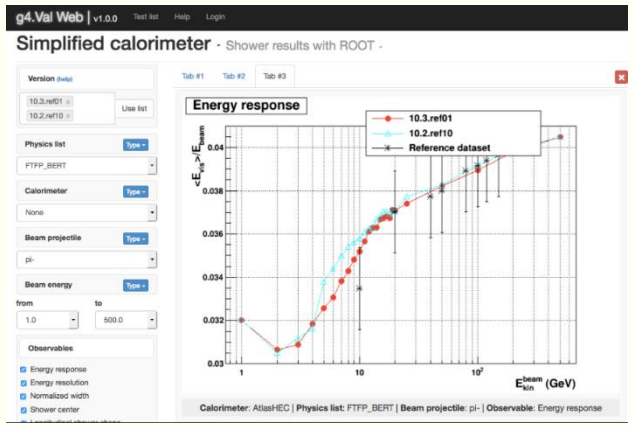
- The CERN/SFT group, which contributes to the development, testing, deployment and support of the toolkit, is also in charge of monthly running a set community-developed tests using the development releases of Geant4.

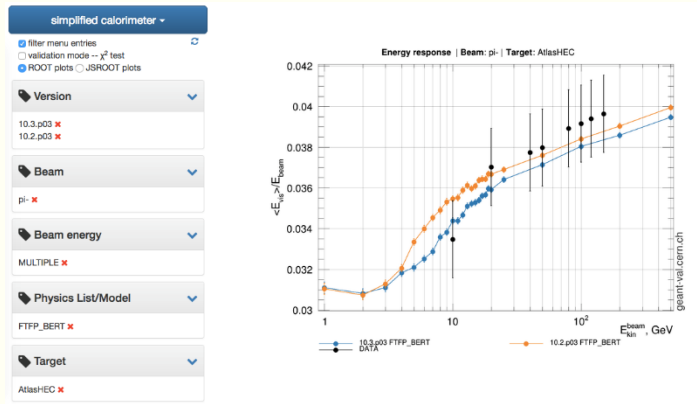
- We developed the Web application "geant-val" for visualizing results of these tests and comparing them between different Geant4 releases. The application is written using Express.js, Node.js and AngularJS framework, and uses PostgreSQL for storing test results.



History

- **2013:** first validation page **g4-val** was created by G.Lestaris (CERN technical student), aimed to facilitate "hadronic physics" validation by CERN Geant4 group based on DJANGO, MySQL.
- **2016:** I.Ifrim (CERN summer student) has developed a prototype **geant-val** based on Node.js and AngularJS as "interface" to postgresql database of **DOSSIER** project.
- **2017:** After extensive development **geant-val** became production web application used by CERN "hadronic" group.
- **2017-2018:** A lot of improvements and many new features.





geant-config-generator

[Step 1] **Clone repo**

```
git clone ssh://git@gitlab.cern.ch:7999/GeantValidation/geant-config-generator.git
Cloning into 'geant-config-generator'...
```

[Step 2] **Submit jobs**

```
./mc-config-generator.py generate -t geant4/test37 -v 10.5.beta01 -q 1nd -r
Prepared 594 jobs for test test37.
Continue sumbitting in batch [y|n]?
Job <192600064> is submitted to queue <1nd>.
Job <192600065> is submitted to queue <1nd>.
Job <192600067> is submitted to queue <1nd>.
.........................
```

[Step 2.1] **Check status**

```
./mc-config-generator.py status -t geant4/test37 -d OUTPUT/
 == Geant4 10.5.beta01
  Test: test37
    Failed: 120
    Running: 296
    Pending: 178
    Total: 594
```

[Step 3] **convert/parse results to input geant-val JSON**

```
./mc-config-generator.py parse -t geant4/test37 -d OUTPUT/
```

[Step 4] **Upload JSONs to geant-val**

```
python geant_upload.py --krb -j OUTPUTJSON/geant4/10.5.beta01/x86_64-slc6-gcc49-opt/test37/*
```

Geant-config-generator

Motivation:

Test are different: different ways to configure and run, different output (ascii files, ROOT files), no axis names, histograms can have name like h11), no titles.

Geant-config-generator is Python-based utility managing user's physics tests (a side product of **geant-val** development)

- facilitates and makes more transparent the creation of test configurations from test configuration template and steering file.
- submits jobs to batch systems(CERN LSF and HTCondor).
- parses and combine produced results.
- Add missing meta information
- Pass info about parameters used from test config to "geant-val"
- creates input JSON files for **geant-val.**

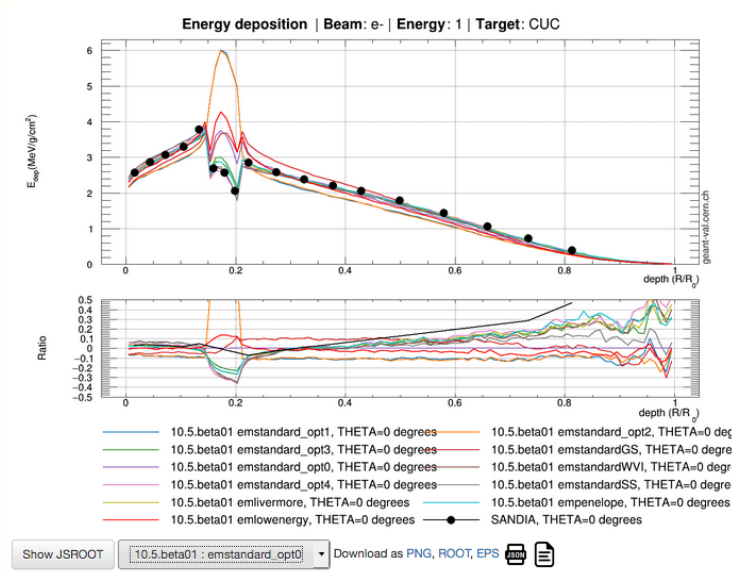
```
#verbose
0
//            proton pi- pi+ kaon+ kaon- anti_proton
#particle
%PARTICLE%
//
#material
%MATERIAL%
#targetA
%TARGETA%
//            ftfp ftfb qgsp qgsb bertini binary
#generator
%GENERATOR%
//
#events
%NEVENTS%
//
//--------
#Plab(GeV/c)
%ENERGY%
#run
#exit
```

params.conf 582 Bytes

```
1  ! PARTICLE=proton, pi+, pi-
2  ! ENERGY=3.0,5.0,8.0,12.0
3  ! GENERATOR=ftfp
4  !CONST:ENERGY_UNIT=GeV
5  PARTICLE | ENERGY | MATERIAL | TARGETA | GENERATOR | NEVENTS
6  PARTICLE | ENERGY |  G4_Pb   | 208     | GENERATOR | 2000000
7  PARTICLE | ENERGY |  G4_Ta   | 181     | GENERATOR | 2000000
8  PARTICLE | ENERGY |  G4_Sn   | 119     | GENERATOR | 2000000
9  PARTICLE | ENERGY |  G4_Al   |  27     | GENERATOR | 2000000
10 PARTICLE | ENERGY |  G4_Cu   |  64     | GENERATOR | 2000000
11 PARTICLE | ENERGY |  G4_C    |  12     | GENERATOR | 2000000
12 PARTICLE | ENERGY |  G4_Be   |   9     | GENERATOR | 2000000
```

What “geant-val” can

- allows visual plots comparison using overlaying and ratio plots
  - Release with release
  - Release with experimental data
  - Physics models/physics lists
- allows to perform statistical comparison of test results
- displays static PNG plots (better quality) and interactive plots using JSROOT
- allows to download plots in PNG, ROOT and EPS formats
- allows to download input histograms in JSON and GNUplot formats

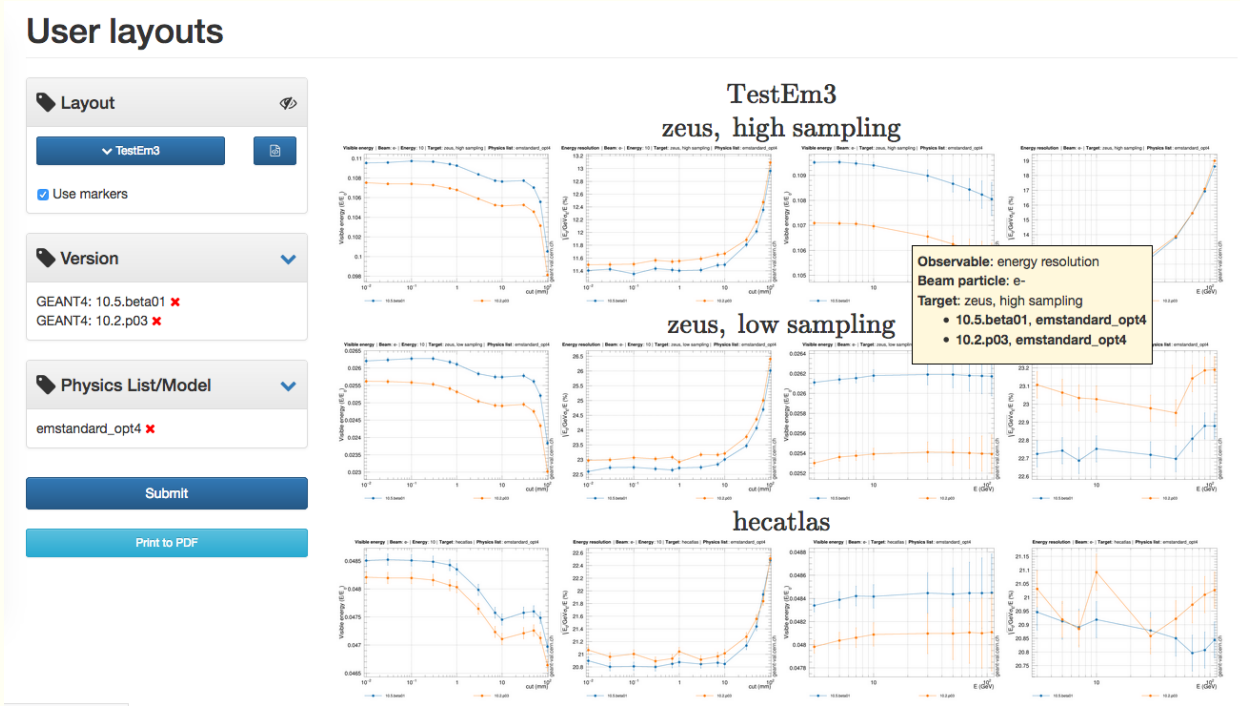


User layouts - https://geant-val.cern.ch/layouts

TestEm3.xml 3.47 KB

Edit | Web IDE | Replace | Delete

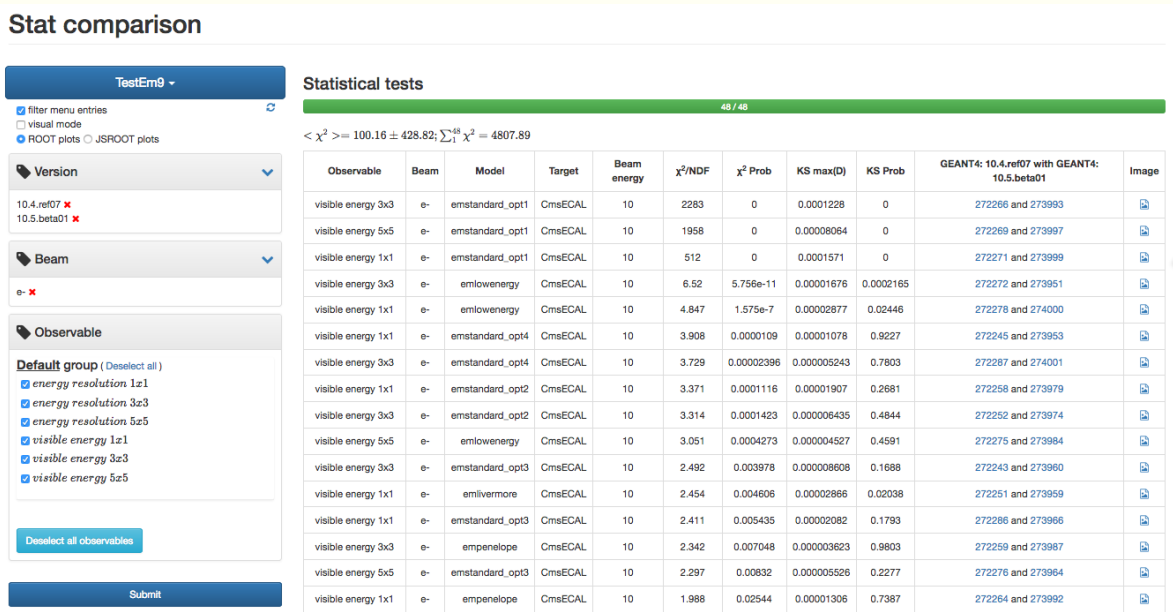
```
<?xml version="1.0" encoding="UTF-8"?>
<layout>
  <default model="emstandard_opt0"></default>
  <row>
  <label text="\LARGE{TestEm3}" colspan="4"/>
  </row>

  <row><label text="\LARGE{zeus, high sampling}" colspan="4"/></row>
  <row>
    <plot test="TestEm3-cutTest" observable="visible energy"    beam="e-" energy="10" secondary="None" target="zeus, high sampling" yaxis="lin" xaxis="log"
    <plot test="TestEm3-cutTest" observable="energy resolution" beam="e-" energy="10" secondary="None" target="zeus, high sampling" yaxis="lin" xaxis="log"
    <plot test="TestEm3-Energy" observable="visible energy"    beam="e-" energy="MULTIPLE" secondary="None" target="zeus, high sampling" yaxis="lin" xaxis="
    <plot test="TestEm3-Energy" observable="energy resolution" beam="e-" energy="MULTIPLE" secondary="None" target="zeus, high sampling" yaxis="lin" xaxis="
  </row>

  <row><label text="\LARGE{zeus, low sampling}" colspan="4"/></row>
  <row>
    <plot test="TestEm3-cutTest" observable="visible energy"    beam="e-" energy="10" secondary="None" target="zeus, low sampling" yaxis="lin" xaxis="log" t
    <plot test="TestEm3-cutTest" observable="energy resolution" beam="e-" energy="10" secondary="None" target="zeus, low sampling" yaxis="lin" xaxis="log" t
    <plot test="TestEm3-Energy" observable="visible energy"    beam="e-" energy="MULTIPLE" secondary="None" target="zeus, low sampling" yaxis="lin" xaxis="l
    <plot test="TestEm3-Energy" observable="energy resolution" beam="e-" energy="MULTIPLE" secondary="None" target="zeus, low sampling" yaxis="lin" xaxis="l
  </row>

  <row><label text="\LARGE{hecatlas}" colspan="4"/></row>
  <row>
    <plot test="TestEm3-cutTest" observable="visible energy"    beam="e-" energy="10" secondary="None" target="hecatlas" yaxis="lin" xaxis="log" title=""/>
    <plot test="TestEm3-cutTest" observable="energy resolution" beam="e-" energy="10" secondary="None" target="hecatlas" yaxis="lin" xaxis="log" title=""/>
    <plot test="TestEm3-Energy" observable="visible energy"    beam="e-" energy="MULTIPLE" secondary="None" target="hecatlas" yaxis="lin" xaxis="log" title=
    <plot test="TestEm3-Energy" observable="energy resolution" beam="e-" energy="MULTIPLE" secondary="None" target="hecatlas" yaxis="lin" xaxis="log" title=
  </row>
```



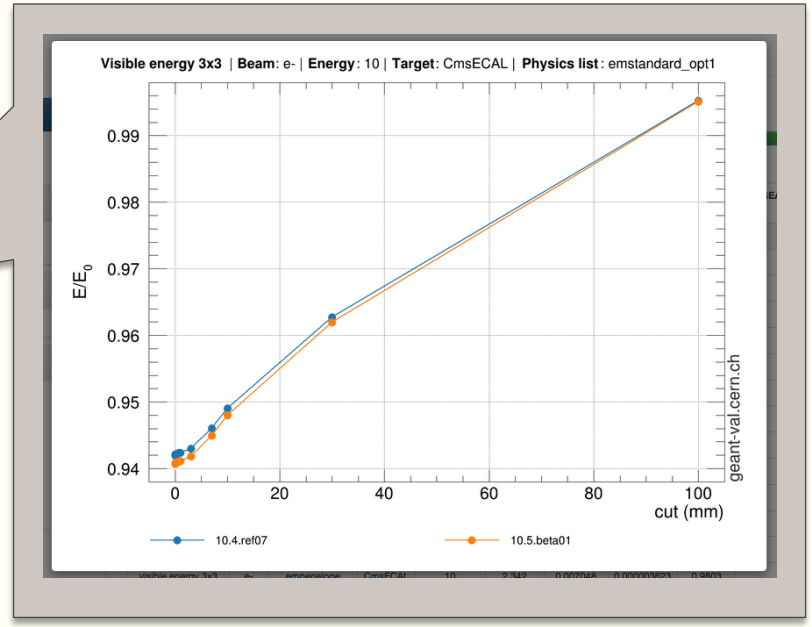
User-defined layouts can be used for fast visual validation of Geant4. User can define their own XML template to show the plots in the layout they want.. Some predefined templates are already available in the application.

Statistical comparison - https://geant-val.cern.ch/stat

χ^2 and Kolmogorov-Smirnov statistical tests allows test results for **different** versions of Geant4 to be compared. The calculations are performed on the client side using JavaScript workers.



Observable	Beam	Model	Target	Beam energy	χ^2/NDF	χ^2 Prob	KS max(D)	KS Prob	GEANT4: 10.4.ref07 with GEANT4: 10.5.beta01	Image
visible energy 3x3	e-	emstandard_opt1	CmsECAL	10	2283	0	0.0001228	0	272266 and 273993	
visible energy 5x5	e-	emstandard_opt1	CmsECAL	10	1958	0	0.00008064	0	272269 and 273997	
visible energy 1x1	e-	emstandard_opt1	CmsECAL	10	512	0	0.0001571	0	272271 and 273999	
visible energy 3x3	e-	emlowenergy	CmsECAL	10	6.52	5.756e-11	0.00001676	0.0002165	272272 and 273951	
visible energy 1x1	e-	emlowenergy	CmsECAL	10	4.847	1.575e-7	0.00002877	0.02446	272278 and 274000	
visible energy 1x1	e-	emstandard_opt4	CmsECAL	10	3.908	0.0000109	0.00001078	0.9227	272245 and 273953	
visible energy 3x3	e-	emstandard_opt4	CmsECAL	10	3.729	0.00002396	0.000005243	0.7803	272287 and 274001	
visible energy 1x1	e-	emstandard_opt2	CmsECAL	10	3.371	0.0001116	0.00001907	0.2681	272258 and 273979	
visible energy 3x3	e-	emstandard_opt2	CmsECAL	10	3.314	0.0001423	0.000006435	0.4844	272252 and 273974	
visible energy 5x5	e-	emlowenergy	CmsECAL	10	3.051	0.0004273	0.000004527	0.4591	272275 and 273984	
visible energy 3x3	e-	emstandard_opt3	CmsECAL	10	2.492	0.003978	0.000008608	0.1688	272243 and 273960	
visible energy 1x1	e-	emlivermore	CmsECAL	10	2.454	0.004606	0.00002866	0.02038	272251 and 273959	
visible energy 1x1	e-	emstandard_opt3	CmsECAL	10	2.411	0.005435	0.00002082	0.1793	272286 and 273966	
visible energy 3x3	e-	empenelope	CmsECAL	10	2.342	0.007048	0.000003623	0.9803	272259 and 273987	
visible energy 5x5	e-	emstandard_opt3	CmsECAL	10	2.297	0.00832	0.000005526	0.2277	272276 and 273964	
visible energy 1x1	e-	empenelope	CmsECAL	10	1.988	0.02544	0.00001306	0.7387	272264 and 273992	



Caveat: often experimental results do not match with MC dots (X coordinates, bin size)

Tests integrated into **geant-val**

Test name	Project	Author	Description
FluctTest	GEANT4	Vladimir Ivanchenko	Example for investigation of G4 fluctuation models is on the base of TestEm8 (ionisation in thin absorbers and gaseous detectors). Simulation data are ... Show more
MscHanson	GEANT4	Vladimir Ivanchenko	The tests simulates multiple scattering distributions of 15.7 MeV electrons transmitted through thin foils of the indicated materials.
TestEm3-Energy	GEANT4	Vladimir Ivanchenko	TestEm3 simulates energy resolution and visible energy in hecatlas, zeus calorimeters
TestEm3-cutTest	GEANT4	Vladimir Ivanchenko	TestEm3 produces plots showing impact of production cut on energy resolution and visible energy
TestEm9	GEANT4	Vladimir Ivanchenko	TestEm9
hadr00	GEANT4	Vladimir Ivanchenko	Application demonstrating Geant4 hadronic cross sections
simplified calorimeter	GEANT4	Andrea Dotti, Alberto Ribon	The simulation program is implemented as a simplified geometry of sampling calorimeters. The calorimeter geometry consists of a cylinder with a radius ... Show more
test15	GEANT4	Alexander Howard	Comparison of Geant4 simulation against the TARC experiment. This test is a validation of the production, transportation and interaction of neutrons ... Show more
test22-HARP	GEANT4	Vladimir Uzhinsky	Double-differential proton production cross sections in the proton momentum range 0 to 8 GeV at angles 0.05 to 0.25 radians from collisions of charged ... Show more
test22-NA49	GEANT4	Vladimir Ivanchenko	
test22-NA61	GEANT4	Vladimir Uzhinsky	double differential Pl+, Pl-, proton production cross section in the laboratory system for p+C interactions at 31 GeV/c in different polar angle ... Show more
test37	GEANT4	Vladimir Ivanchenko	test37
test46	GEANT4	Vladimir Ivanchenko	Simulation of simplified combined calorimeter. Crystal electromagnetic (ECAL) calorimeter and sampling hadronic calorimeter (HCAL) are ... Show more
tileatlas	GEANT4	Vladimir Ivanchenko	tileatlas
main02	Pythia8		

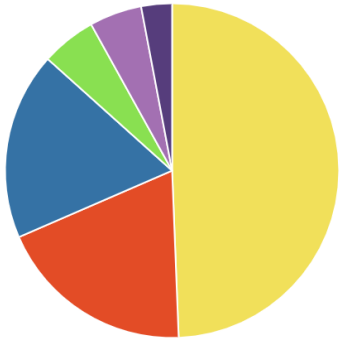
"Integrated" – test can be run and results can be obtained using geant-config-generator

geant-val repository statistics

geant-val

Programming languages used in this repository

JavaScript	49.37 %
HTML	19.12 %
Python	18.13 %
Shell	5.31 %
PLpgSQL	5.08 %
CSS	2.98 %



Commit statistics for master Aug 30 - Aug 26

master

GVP

- Total: **1699 commits**
- Average per day: **2 commits**
- Authors: **21**

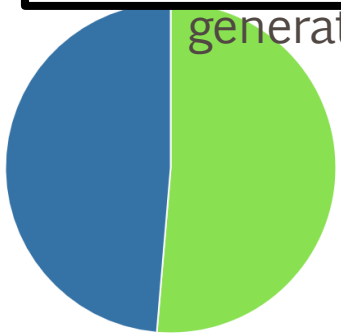
Commits per day of month

100

geant-config-generator

Programming languages used in this repository

Shell	51.34 %
Python	48.66 %



Commit statistics for master Oct 09 - Aug 22

master

geant-config-generator

- Total: **461 commits**
- Average per day: **1 commits**
- Authors: **4**

Commits per day of month

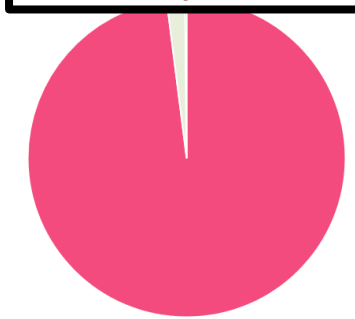
50

40

plotter

Programming languages used in this repository

C++	97.98 %
CMake	1.85 %
Shell	0.16 %



Commit statistics for master Mar 21 - Aug 26

master

plotter

- Total: **294 commits**
- Average per day: **0 commits**
- Authors: **8**

Commits per day of month

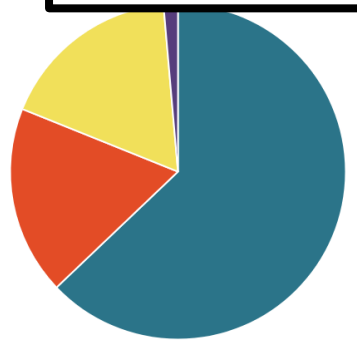
30

25

R&D (new)

Programming languages used in this repository

TypeScript	62.88 %
HTML	18.21 %
JavaScript	17.53 %
CSS	1.38 %



Commit statistics for master Jul 16 - Aug 27

master

Loopback

- Total: **397 commits**
- Average per day: **9 commits**
- Authors: **5**

Commits per day of month

50

40

Ongoing activities

Motivation:

- In January 2018, a schedule was announced for phasing-out AngularJS: after releasing 1.7.0, the active development on AngularJS will continue till June 30, 2018. Afterwards, 1.7 will be supported till June 30, 2021 as long-term support.

- The backend was originally developed for the simplified calorimeter test:
  - The parameter names are hardcoded in the database schema.
  - Workarounds are sometimes needed to implement other tests.
  - "hard-coded" parameters in database schema ("beam particle", "beam energy", "model", "secondary particle") limit web app usage to "fixed target" tests.

But we would like to extend geant-val functionality to exclude MC simulation results (Pythia8, MC@NLO etc)

R&D:

- To try "parameter free" db schema

- To get experience with Angular2+

Conclusion

We have developed a web application **geant-val** facilitating validation of Geant4 which have **intuitive user interface, nice graphics**

The app provides:

- **consistent storage of test results**
- **overlaying plots and ratio plots for regression testing**
- **possibility for comparison with experimental data**
- **simple statistical evaluation for regression testing**

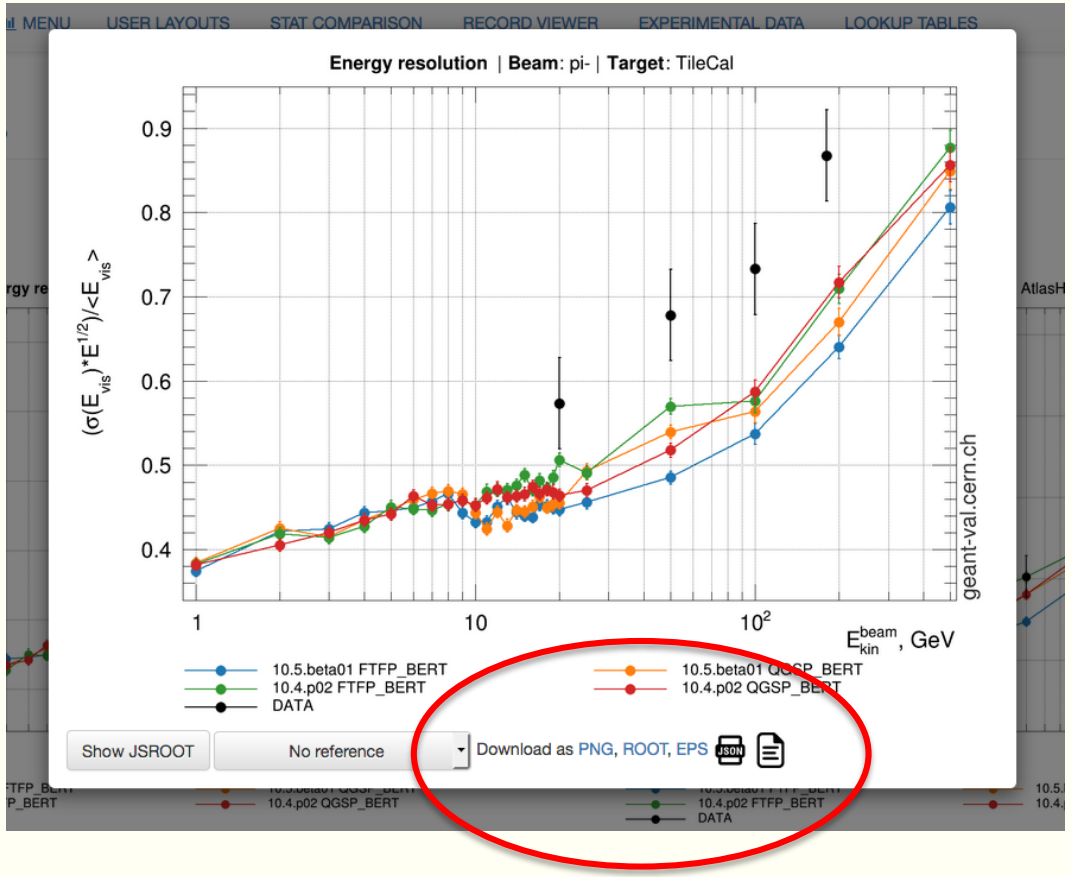
The geant-val works and used by CERN Geant4 team.

We are working on new version of **geant-val** which **will try to accommodate more generic cases and our current understanding of validation aspects.**

Backup

- Backup slides

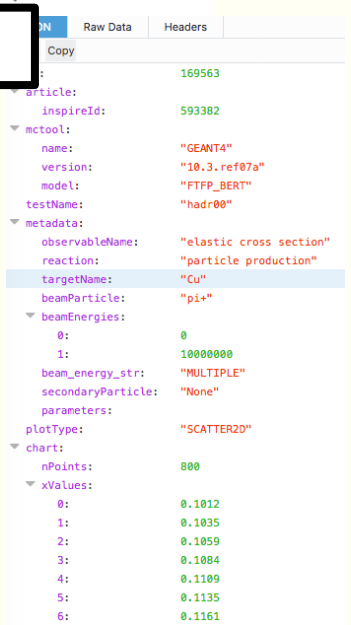
data download formats



GNUPlot

```
#! /usr/bin/gnuplot -persist
# ID:                   274682
# Test:                 simplifi
# Tool:                 GEANT4 1
# Beam:                 pi-
# Beam Energy:          Multiple
# Observable:           energy resolution
# Secondary:            None
# Target:               TileCal
# Parameters:
set term png size 1280,1024
set output              "274682.png"

set title "Beam: pi-,Energy: Multiple,Target: TileCal"
set term png
set xlabel "E_{kin}^{beam}, GeV"
set ylabel "(\sigma(E_{vis}))*E^{1/2})/<E_{vis}>"
set bars small
set grid
plot '-' using 1:2:($1-sqrt($3**2+$7**2)):($1+sqrt($4**2+$8**2)):($2-sqrt($5**2+$9**2)):($2+sqrt($6**2+$10**2)) notitle with xyerrorlines linecolor rgb "blue"
XVALUE YVALUE   XSTATERRORMINUS XSTATERRORPLUS YSTATERRORMINUS YSTATERRORPLUS XSYSERRORMINUS XSYSERRORPLUS YSYSERRORMINUS YSYSERRORPLUS
1       0.374314 0             0             0.00657689    0.00657689    0             0             0             0
2       0.421365 0             0             0.00764748    0.00764748    0             0             0             0
3       0.424342 0             0             0.00740783    0.00740783    0             0             0             0
4       0.443874 0             0             0.00740141    0.00740141    0             0             0             0
5       0.445425 0             0             0.00737465    0.00737465    0             0             0             0
6       0.449508 0             0             0.0072578     0.0072578     0             0             0             0
7       0.457138 0             0             0.00709958    0.00709958    0             0             0             0
8       0.466342 0             0             0.00770819    0.00770819    0             0             0             0
9       0.443578 0             0             0.00734782    0.00734782    0             0             0             0
10      0.432816 0             0             0.00705248    0.00705248    0             0             0             0
11      0.432845 0             0             0.00735374    0.00735374    0             0             0             0
12      0.451215 0             0             0.00706221    0.00706221    0             0             0             0
13      0.462203 0             0             0.00870287    0.00870287    0             0             0             0
14      0.444232 0             0             0.00777311    0.00777311    0             0             0             0
15      0.439876 0             0             0.0067379     0.0067379     0             0             0             0
16      0.438588 0             0             0.00676719    0.00676719    0             0             0             0
17      0.45308  0             0             0.00719868    0.00719868    0             0             0             0
18      0.451688 0             0             0.0074046     0.0074046     0             0             0             0
19      0.449497 0             0             0.00759998    0.00759998    0             0             0             0
20      0.447449 0             0             0.00786784    0.00786784    0             0             0             0
25      0.455899 0             0             0.00801242    0.00801242    0             0             0             0
50      0.485307 0             0             0.00795515    0.00795515    0             0             0             0
100     0.537198 0             0             0.0124221     0.0124221     0             0             0             0
200     0.640052 0             0             0.0140328     0.0140328     0             0             0             0
500     0.806039 0             0             0.0197997     0.0197997     0             0             0             0
```

JSON format



- JSON output = JSON input format

Validation workflow

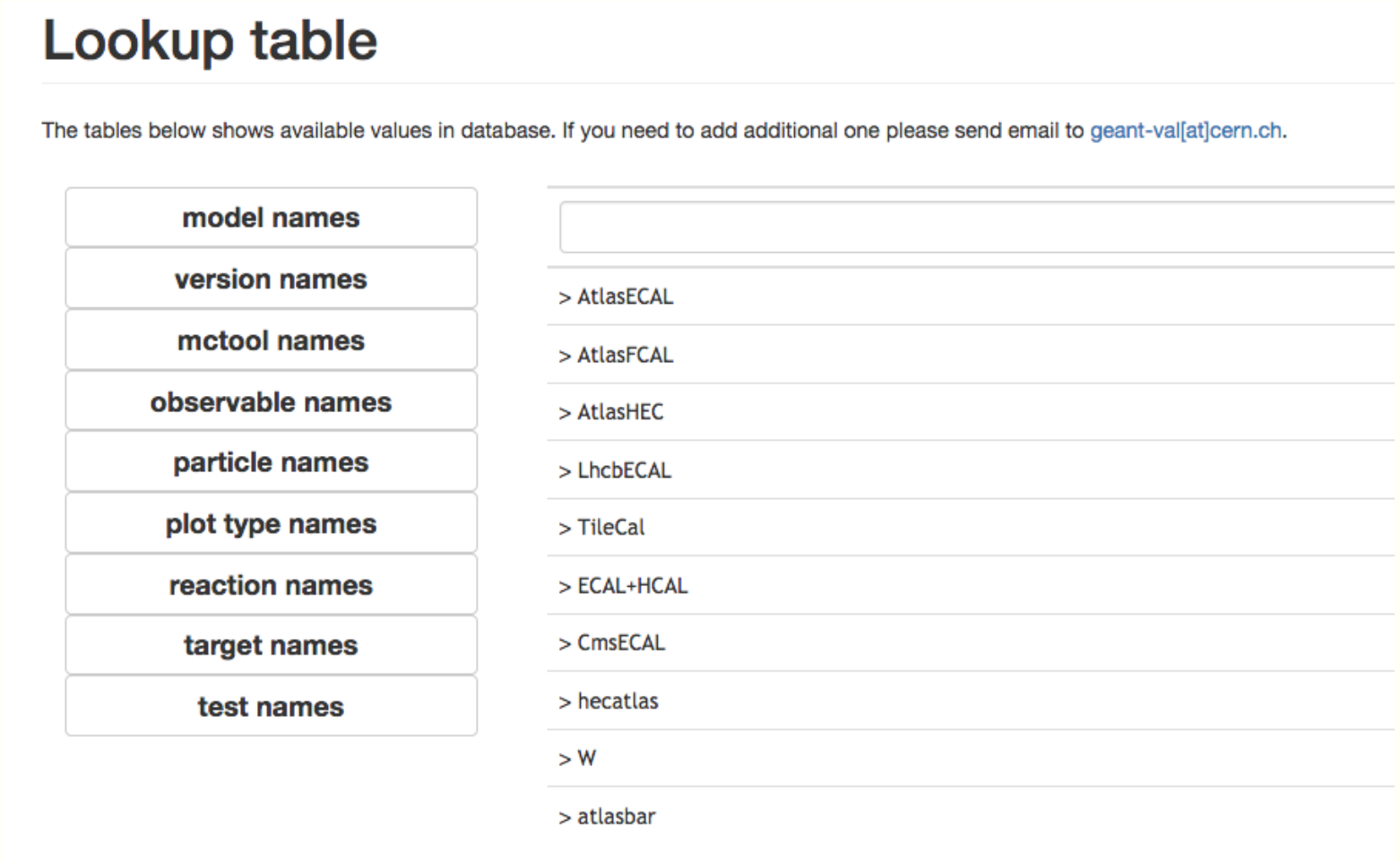
① Run test code locally, using batch systems or with GRID.

② Convert/combine resulted histograms (ascii, ROOT) into **geant-val** JSON objects.

③ Upload JSONs using a dedicated python script into **geant-val**

④ Use **geant-val.cern.ch** – regression testing, statistical comparison

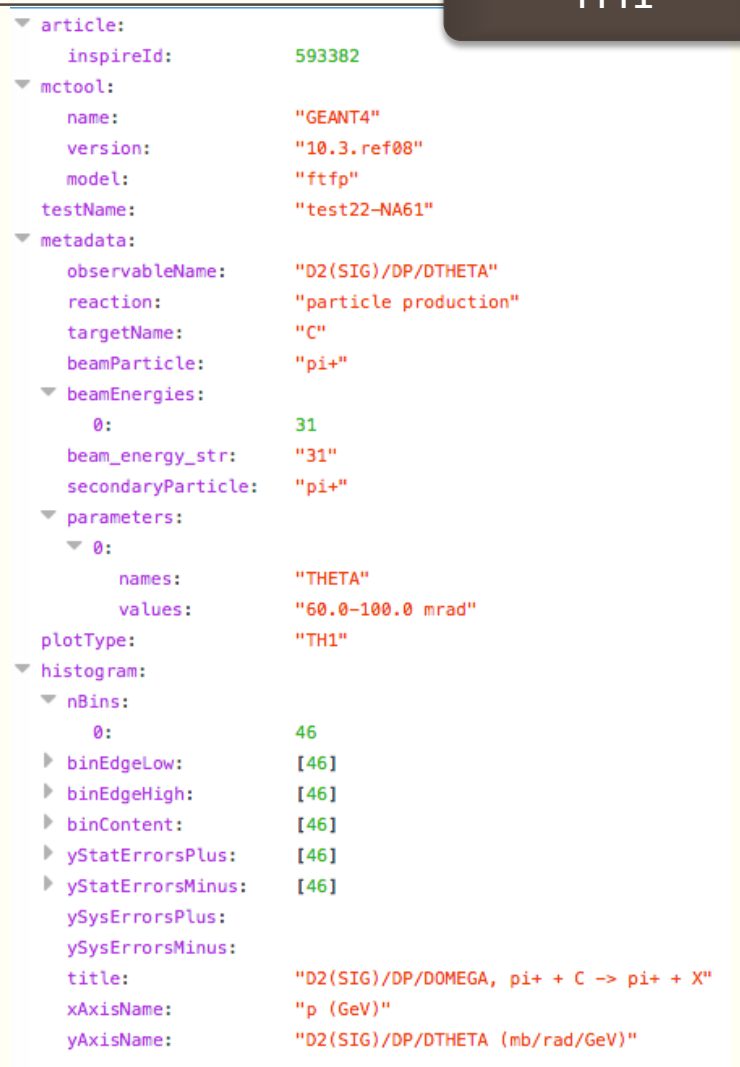
Validation table



Input JSON format

- Simple representation of histogram and corresponding metadata describing configuration/conditions.
- Human readable
- NB: metadata are kept in DB validation tables implement data integrity.

TH1



TGraphErrors

First step towards statistical evaluation

test22-HARP produces ~700 plots per Geant4 release

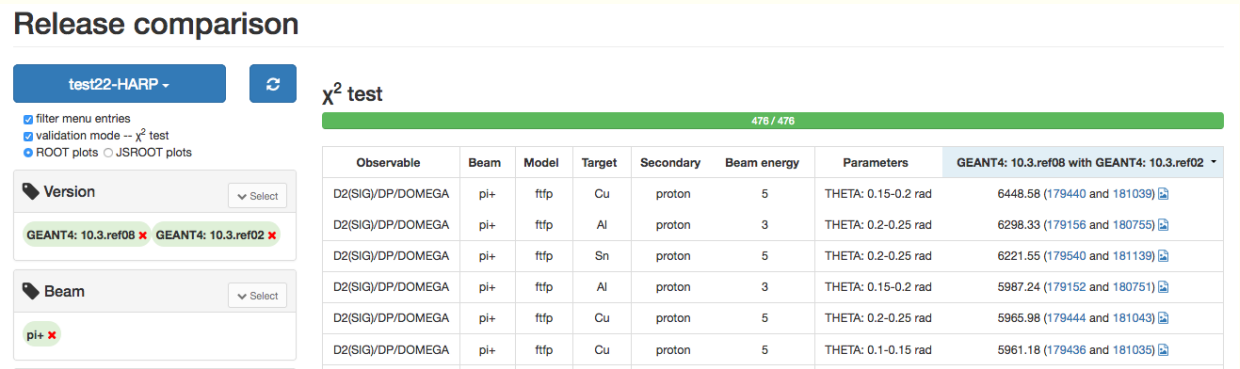
How to compare them with previous release of with experimental data?

How to organize plots for reliable validation?

Introduction of statistical analysis is important!

Possibility to generate table for chi2/ndf between two Geant versions added.

ordering by chi2 value to see histograms with more prominent difference



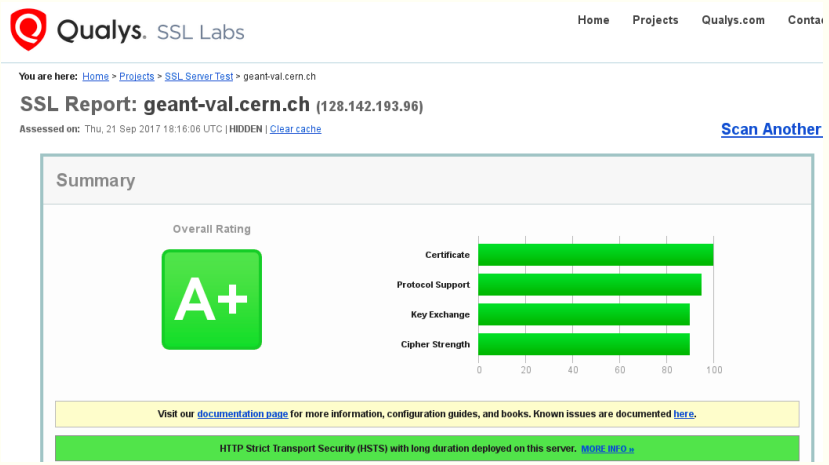
Security

Host security:

- code builds and is encapsulated in docker image
- server process runs as unprivileged user inside docker container
- SSH access to host domain is allowed only by CERN Kerberos ticket

Web security:

- enforced https protocol
- SSL certificates re-issued every 90 days
- A+ (maximal) SSL security level by Qualys SSL scanner

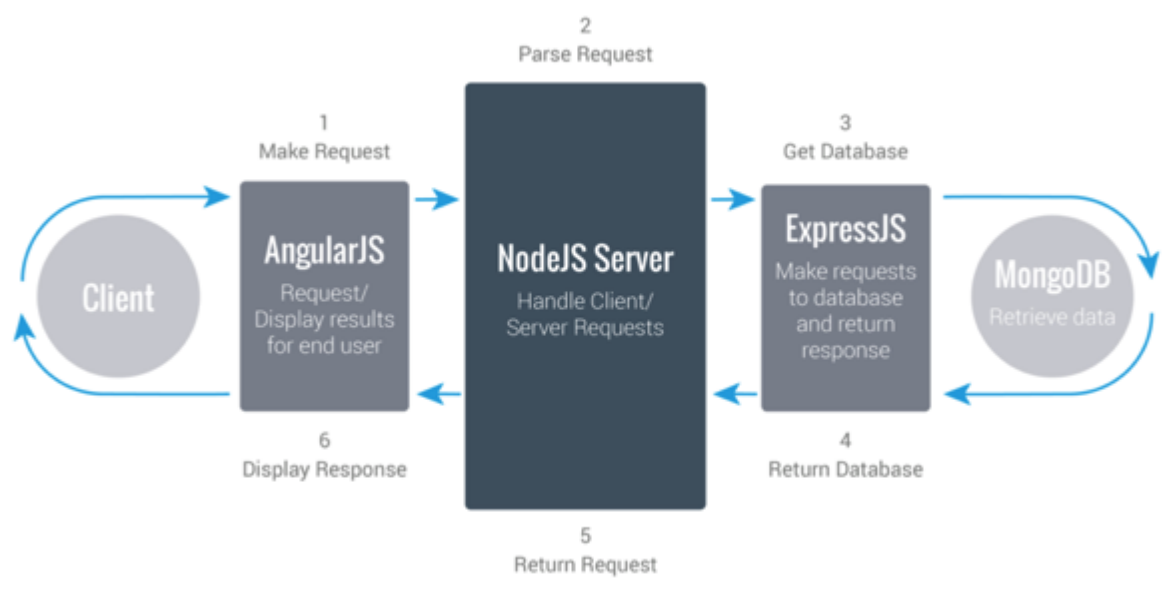


Database security:

- All web APIs scanned for possible SQL injection holes
- Uploading data is allowed only with authorized Kerberos ticket (no other ways to modify data in database)
- Daily database back-up provided by CERN db-on-demand service.

Webserver Architecture

- **Node.js** is an open-source, cross-platform JavaScript run-time environment for executing JavaScript code server-side
- Angular.js is an open-source JavaScript web framework that facilitates the creation of single-page applications and data-driven apps.
- **Express.js** is a web application framework for Node.js
- And instead of MongoDB we use PostrgreSQL



- ✓ Scalability
- ✓ Short development cycles
- ✓ Performance

Access control / CERN SSO authentication

a) for JSON upload with WebAPI

- Only users belonging to geant-val admin e-group

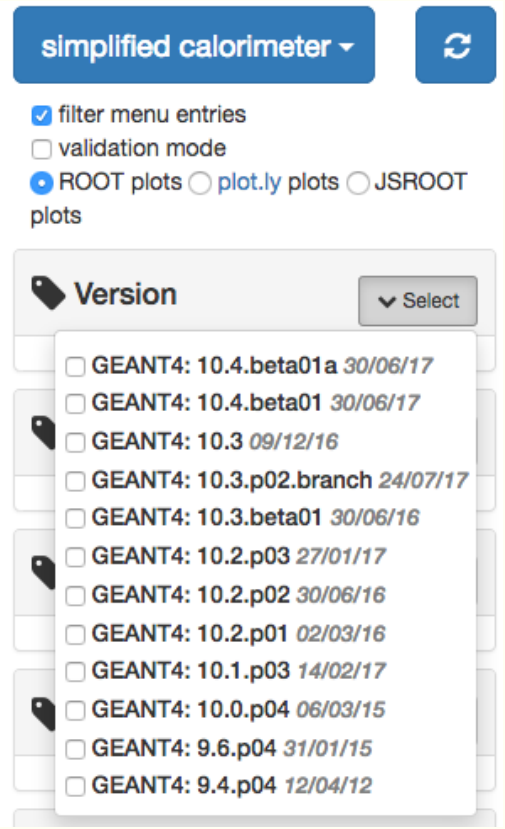
```
python geant_upload.py --krb -j <json file>
```

Alternatively, one can use GRID certificate

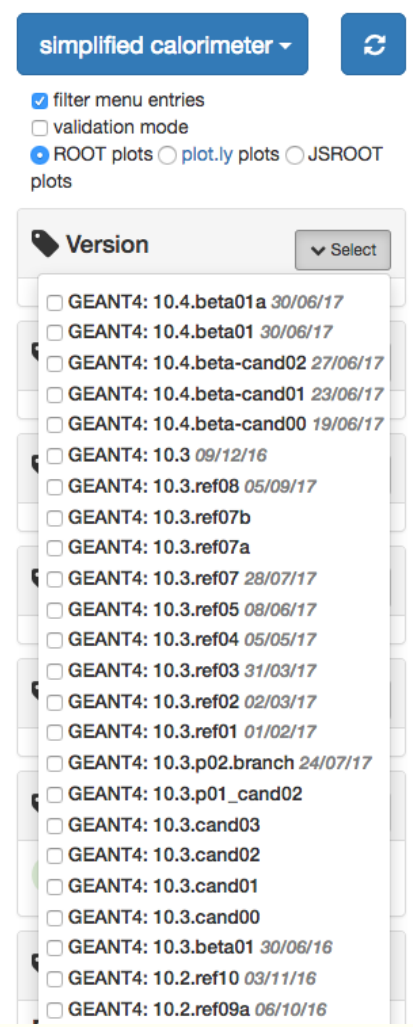
```
python geant_upload.py --key <key file> --cert <cert file> -j <json file>
```

b) restrict access to certain information:

- reference releases
- unvalidated tests



Non authorized view



Authorized view

Problems related to compiling and running of tests

There are many tests written and maintained by **different** developers:

- **different** repos
- **different** compilation ways
- **different** ways to configure and run

Many tests are compiled and placed to CVMFS by Gunter (a lot of work, many thanks!)

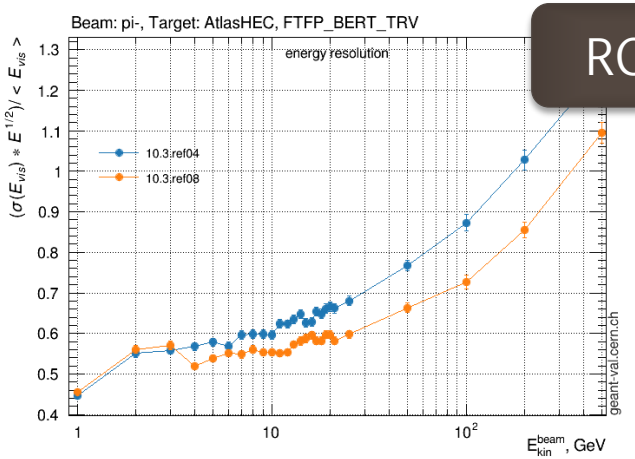
Graphics/Plots

ROOT plots JSROOT plots

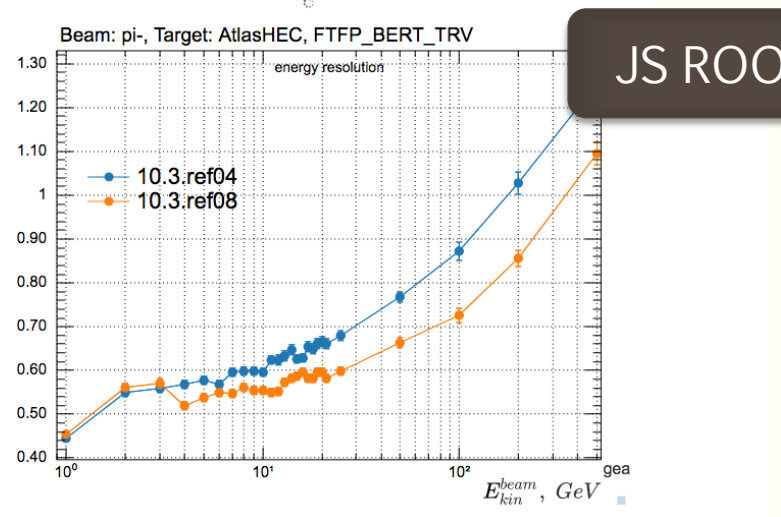
Since last Geant4 collaboration meeting we:

- added static ROOT plots
  - png image created **“on the fly” by c++ code** using ROOT 6
  - small size, quick plotting, cached.
- added JavaScript ROOT plots (JSROOT)
  - JS object created “on the fly” by c++ code using JSROOT
  - bigger object but interactive
- Latex formula renderer moved from MathJax to KaTeX:
  - Much much faster! (around 25 times faster)
  - KaTeX only provides a limited subset of the functionality provided by MathJax, but for our purposes it is enough

ROOT png

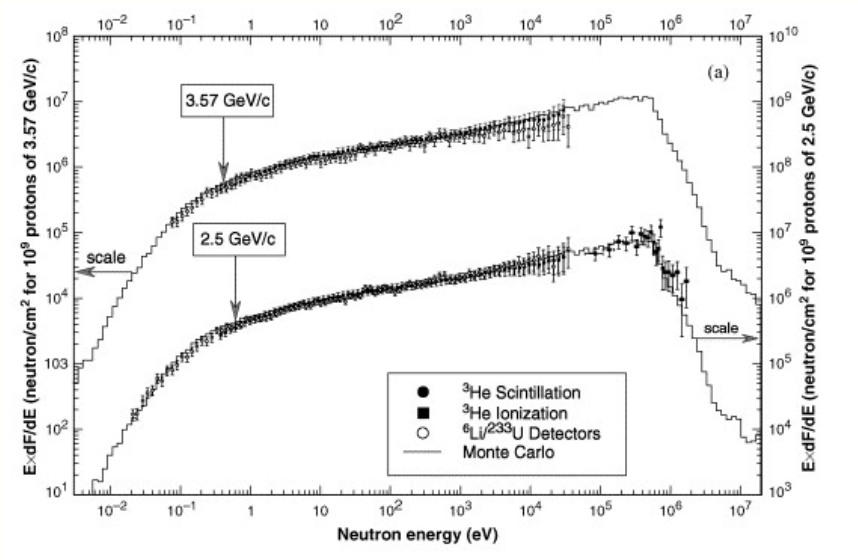


JS ROOT



test15 problem

- **Author(s)** : Alexander Howard
- **Description**: Comparison of Geant4 simulation against the TARC experiment. This test is a validation of the production, transportation and interaction of neutrons with a large (~3.3m) volume of lead. Protons with momenta 2.5GeV/c are impinged on the lead volume through a semi-blind hole and the resulting neutron fluences are measured at various radii within the volume.



Problem:

1) There is no "detector" in experimental data but different methods of measurements.

2) Several experimental measurements describe the same MC simulation.

DB schema and FE are not designed for such use cases (!)

Still can be "inserted" and "plotted" but with some "crazy" workarounds.