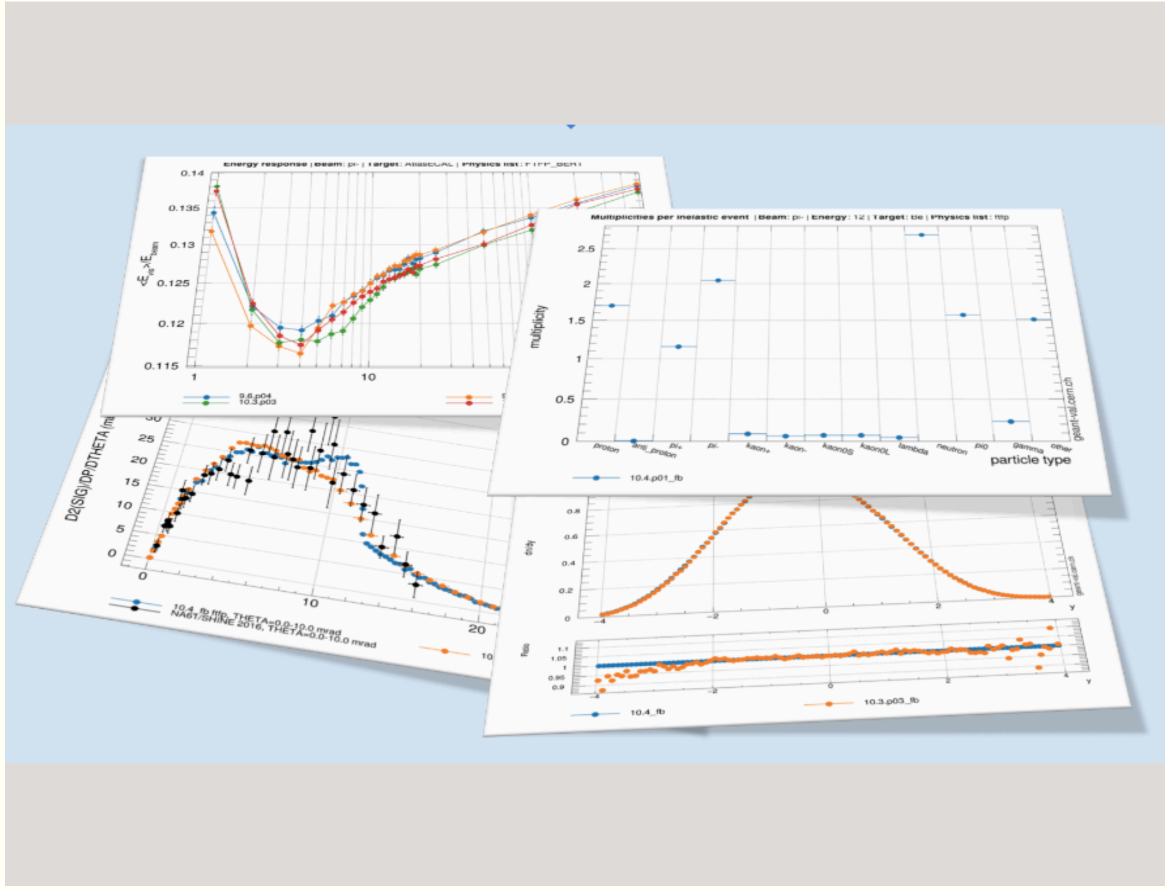
GEANT-VAL: VALIDATION WEB APPLICATION

Dmitri Konstantinov IHEP, Protvino
Grigory Latyshev IHEP, Protvino

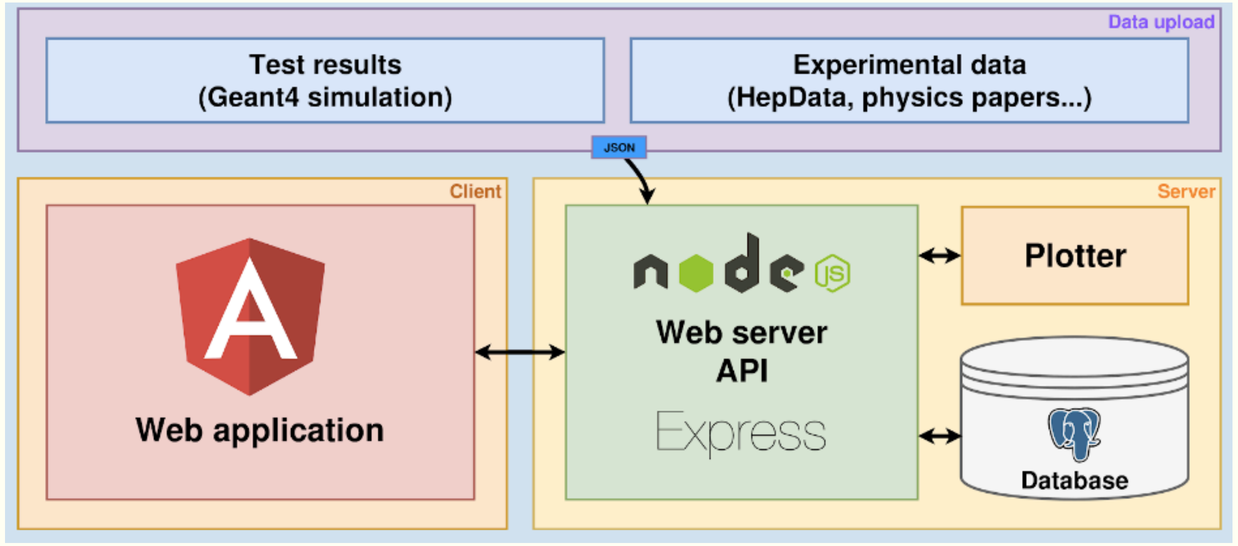
With participation of:
Witold Pokorski, Alberto Ribon, Ivan Razumov, Ioana Ifrim, Luc Feyermuth, Gonzalo De La Cruz, George Lestaris, Hans Wenzel, Julia Yarba.

Virtual Geant4 Collaboration Meeting, 2020



Introduction

- Validation is essential for successful Geant4 development!
- 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 Angular2(v10) framework, and uses PostgreSQL for storing test results.



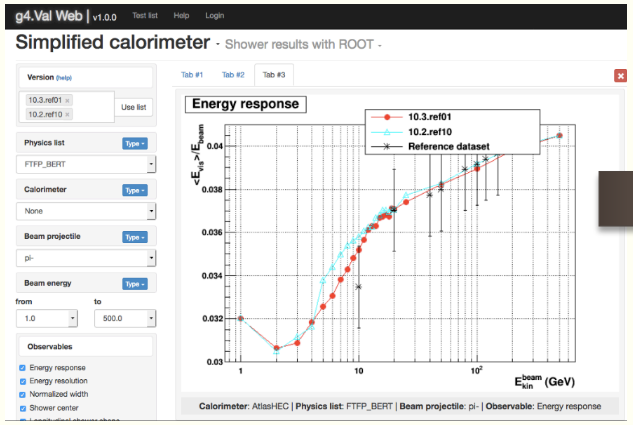
Disclaimer – geant-val can be very slow.

- All “geant-val” plots are produced with ROOT on server side.
- ROOT plots are neat and ROOT gives us interactive JSROOT plots for free.
- ROOT: 200 ms - 1.5 s to produce one PNG file with plots. More overlayed plots – more time to create PNG (?)
- GNUPlot ~ 80 ms per PNG, no matter how many overlaying plos you have.

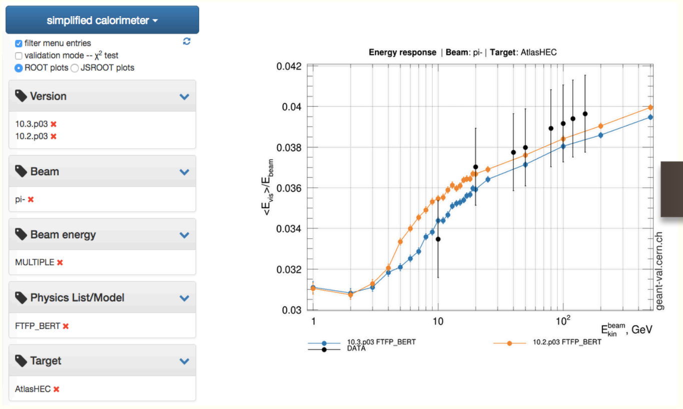
Not critical for now (not so many users) but we are looking for a solution.

History

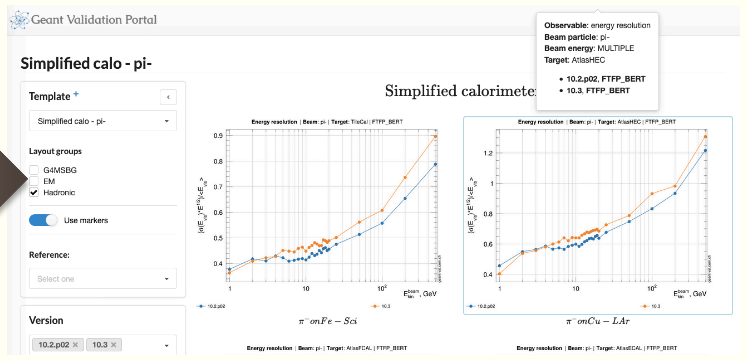
- **2013:** first validation page **g4-val** was created by George Lestaris (CERN technical student), aimed to facilitate “hadronic physics” validation by CERN Geant4 group based on DJANGO, MySQL.
- **2015:** Dmitri Konstantinov, Grigory Latyshev: use-case and requirements studies, development of new DB schema
- **2016:** Iona Ifrim, Dmitri Konstantinov, Grigory Latyshev have developed a prototype of validation page **geant-val** based on Node.js and AngularJS on top of a POSTGRESQL database (developed together with FNAL team – Hans, Julia and Krzysztof.
- **2017:** **geant-val** achieved production quality and became a web application used by CERN “hadronic” group.
- **2017-2018:** A lot of improvements and many new features.
- **2019-2020:** New look, new web framework is used: moved to Angular10 and to FomanticUI from BootstrapUI.


g4.Val Web | v1.0.0
Simplified calorimeter · Shower results with ROOT
Energy response
10.3.ref01
10.2.ref10
Reference dataset
Calorimeter: AtlasHEC | Physics list: FTFP_BERT | Beam projectile: pi- | Observable: Energy response

g4-val



geant-val: AngularJS + Bootstrap



geant-val: Angular2 + FomanticUI

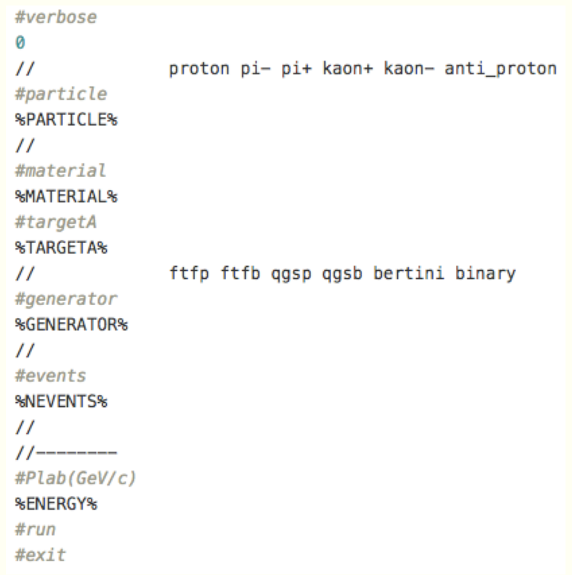
Geant-config-generator – important part!

Motivation:

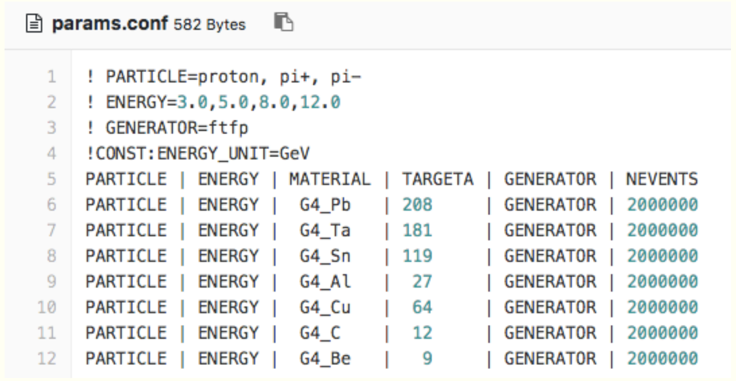
Test are different: different ways to configure and run, different output (ascii files, ROOT files), no axis names, histograms can have name like h11), no titles.

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 system(HTCondor).
- parses and combines produced results.
- add missing meta information
- pass info about parameters used from test config to "geant-val"
- creates input JSON files for **geant-val**.



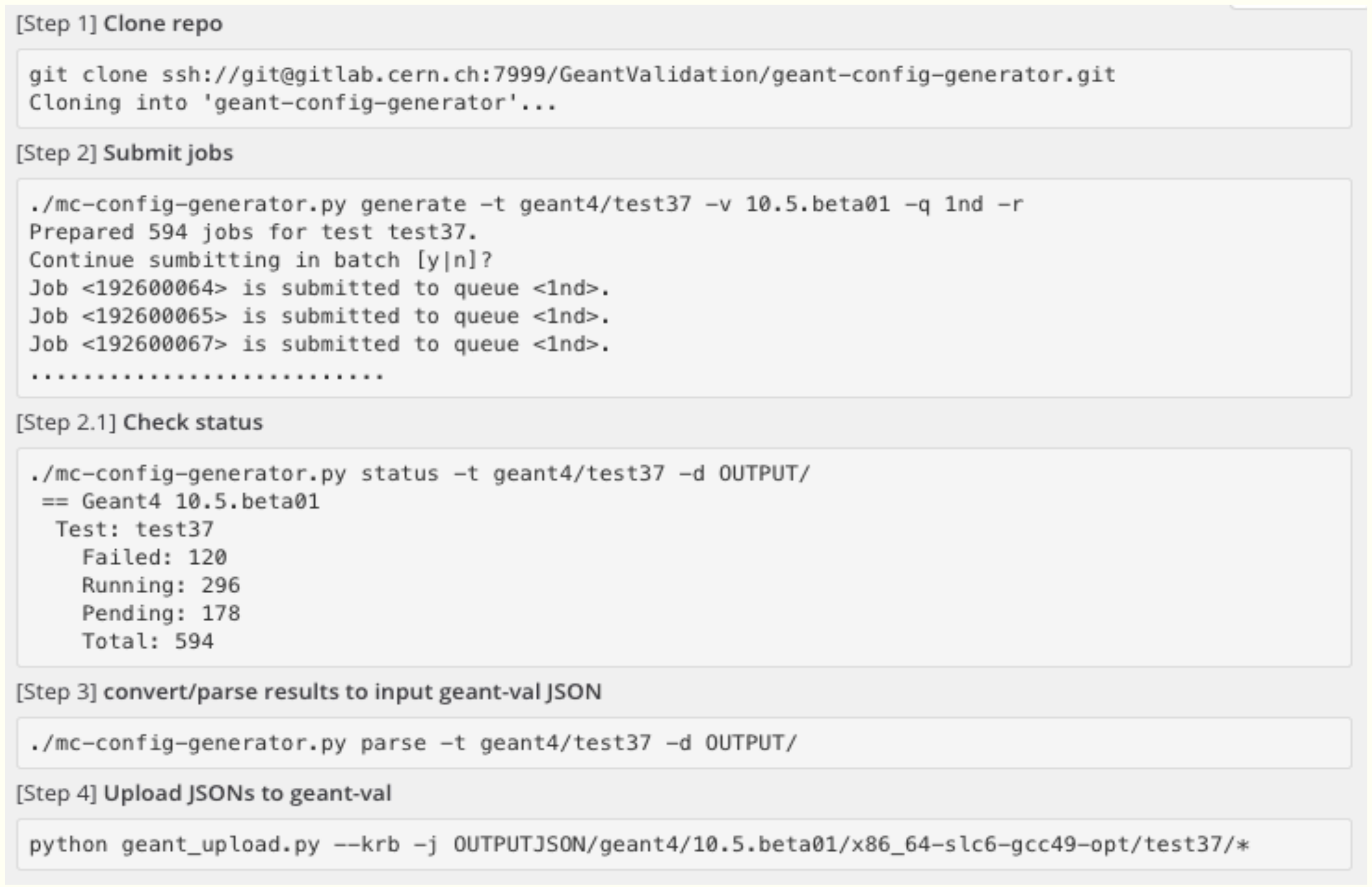
```
#verbose
0
//              proton pi- pi+ kaon+ kaon- anti_proton
#particle
%PARTICLE%
//
#material
%MATERIAL%
#targetA
%TARGETA%
//              ftfp ftfb qgsp qgsb bertini binary
#generator
%GENERATOR%
//
#events
%NEVENTS%
//
//--------
#Plab(GeV/c)
%ENERGY%
#run
#exit
```



params.conf 582 Bytes

```
! PARTICLE=proton, pi+, pi-
! ENERGY=3.0,5.0,8.0,12.0
! GENERATOR=ftfp
!CONST:ENERGY_UNIT=GeV
PARTICLE | ENERGY | MATERIAL | TARGETA | GENERATOR | NEVENTS
PARTICLE | ENERGY |  G4_Pb   | 208     | GENERATOR | 2000000
PARTICLE | ENERGY |  G4_Ta   | 181     | GENERATOR | 2000000
PARTICLE | ENERGY |  G4_Sn   | 119     | GENERATOR | 2000000
PARTICLE | ENERGY |  G4_Al   |  27     | GENERATOR | 2000000
PARTICLE | ENERGY |  G4_Cu   |  64     | GENERATOR | 2000000
PARTICLE | ENERGY |  G4_C    |  12     | GENERATOR | 2000000
PARTICLE | ENERGY |  G4_Be   |   9     | GENERATOR | 2000000
```

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/
```

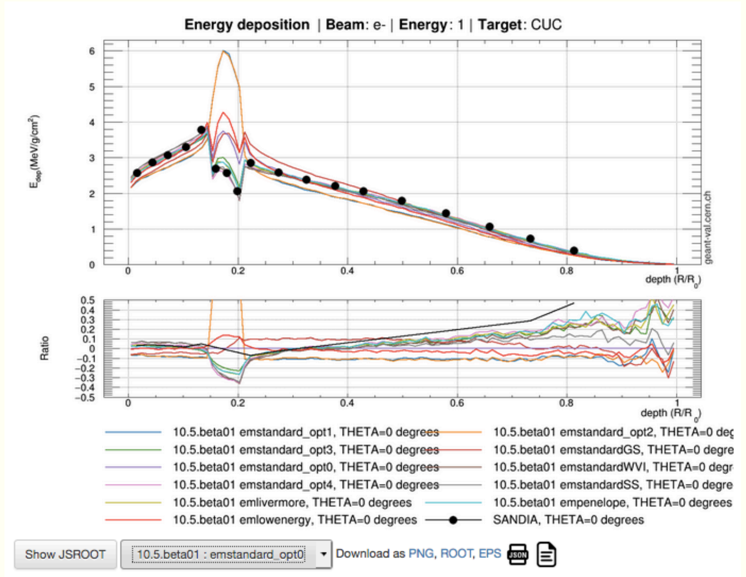
[Step 4] **Upload JSONs to geant-val**

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

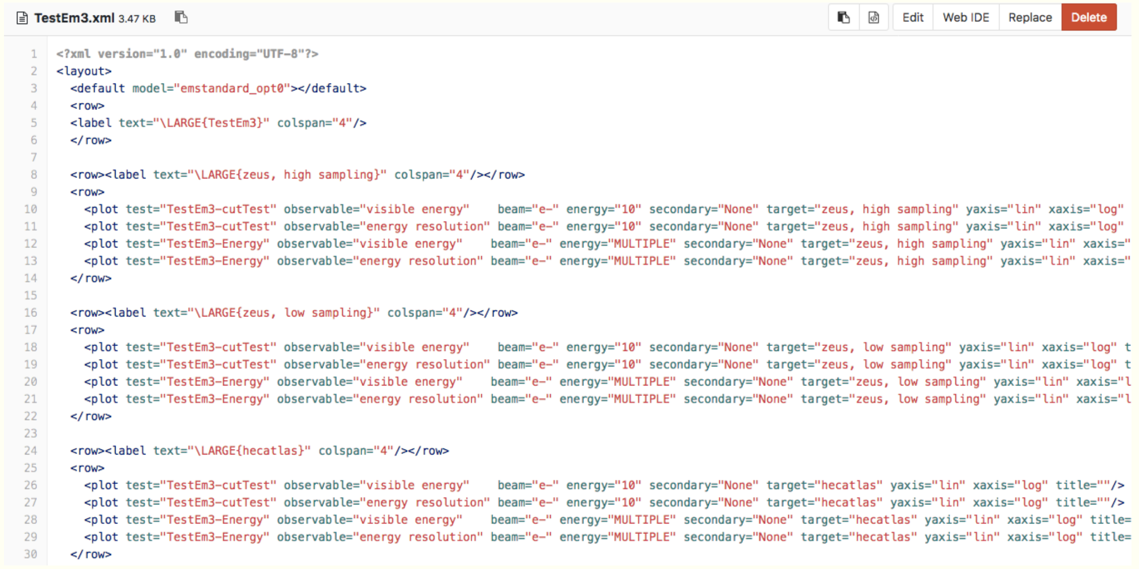
Use **geant-val.cern.ch** – regression testing, statistical comparison

What “geant-val” can do

- 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 (experimental)
- 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


Energy deposition | Beam: e- | Energy: 1 | Target: CUC
E_{dep} (MeV/g/cm^2)
depth (R/R$_0$)
Ratio
geant-val.cern.ch
10.5.beta01 emstandard_opt1, THETA=0 degrees
10.5.beta01 emstandard_opt2, THETA=0 deg
10.5.beta01 emstandard_opt3, THETA=0 degrees
10.5.beta01 emstandardGS, THETA=0 degre
10.5.beta01 emstandard_opt0, THETA=0 degrees
10.5.beta01 emstandardWVI, THETA=0 degr
10.5.beta01 emstandard_opt4, THETA=0 degrees
10.5.beta01 emstandardSS, THETA=0 degre
10.5.beta01 emlivermore, THETA=0 degrees
10.5.beta01 empenelope, THETA=0 degrees
10.5.beta01 emlowenergy, THETA=0 degrees
SANDIA, THETA=0 degrees
Show JSROOT
10.5.beta01 : emstandard_opt0
Download as PNG, ROOT, EPS

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

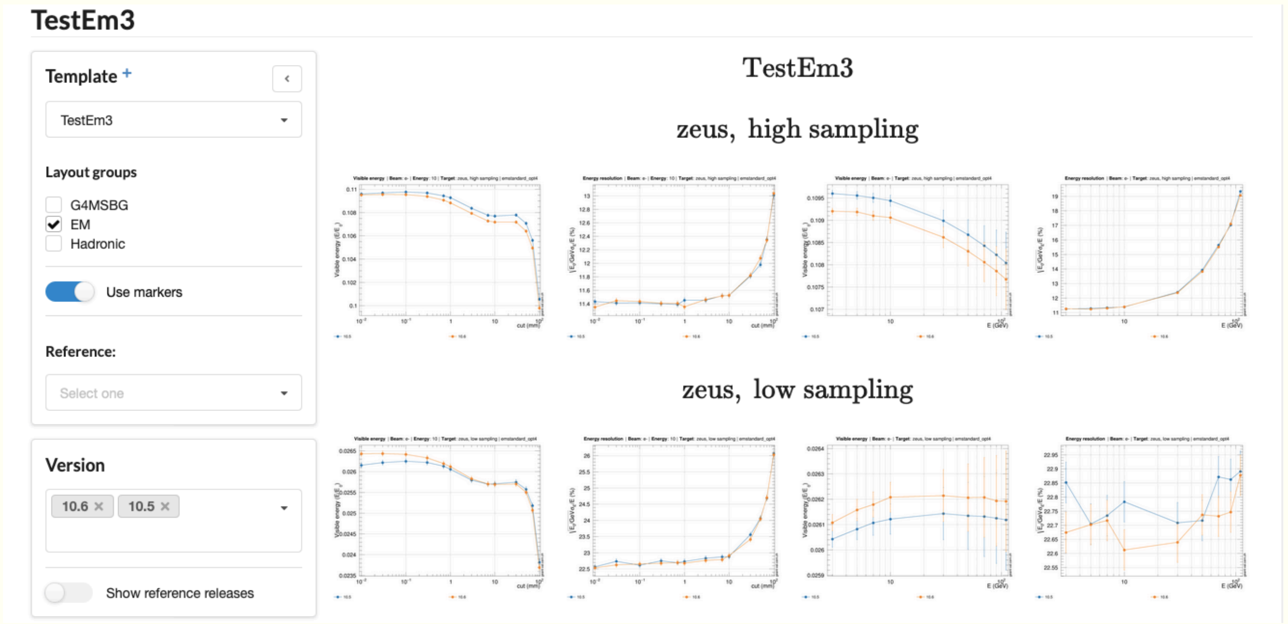


```
<?xml version="1.0" encoding="UTF-8"?>
<layout>
  <default model="emstandard_opt0"></default>
  <row>
  <label text="\LARGE{TestEm3}" colspan="4"/>
  </row>

  <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>
```



TestEm3

Template: TestEm3

Layout groups: G4MSBG [✓ EM] Hadronic

Use markers: on

Reference: Select one

Version: 10.6, 10.5

Show reference releases: off

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.

Tests integrated into **geant-val**

“Integrated” – test can be run, and results can be obtained using geant-config-generator

- “hadrionic physics” tests – used by Alberto R. for validation of monthly reference releases.
- “em” tests - used by Vladimir I. and Mihaly N for electromagnetic validation.
- “G4Med” tests used by medical group (Susanna G., Pedro A. & Co), run for stable releases and for betas.

Discussions/Suggestions

Need for common **neat** repository for Geant4 tests:

- Giving possibility to build easily all tests for given Geant4 release (even for old ones, tagging?)
- Versioning
- Well documented (1 page describing what the test does, what we validate, why, etc)
- With justified number of events and points for validation.
- Defined regularity of running
- Who check validations results and report
- Follow some standards?

We need your feedback! (rather healthy criticism!)

More users – better product!

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 several Geant4 teams.

Several bugs were found with geant-val in reference releases this year (Alberto & Vladimir)

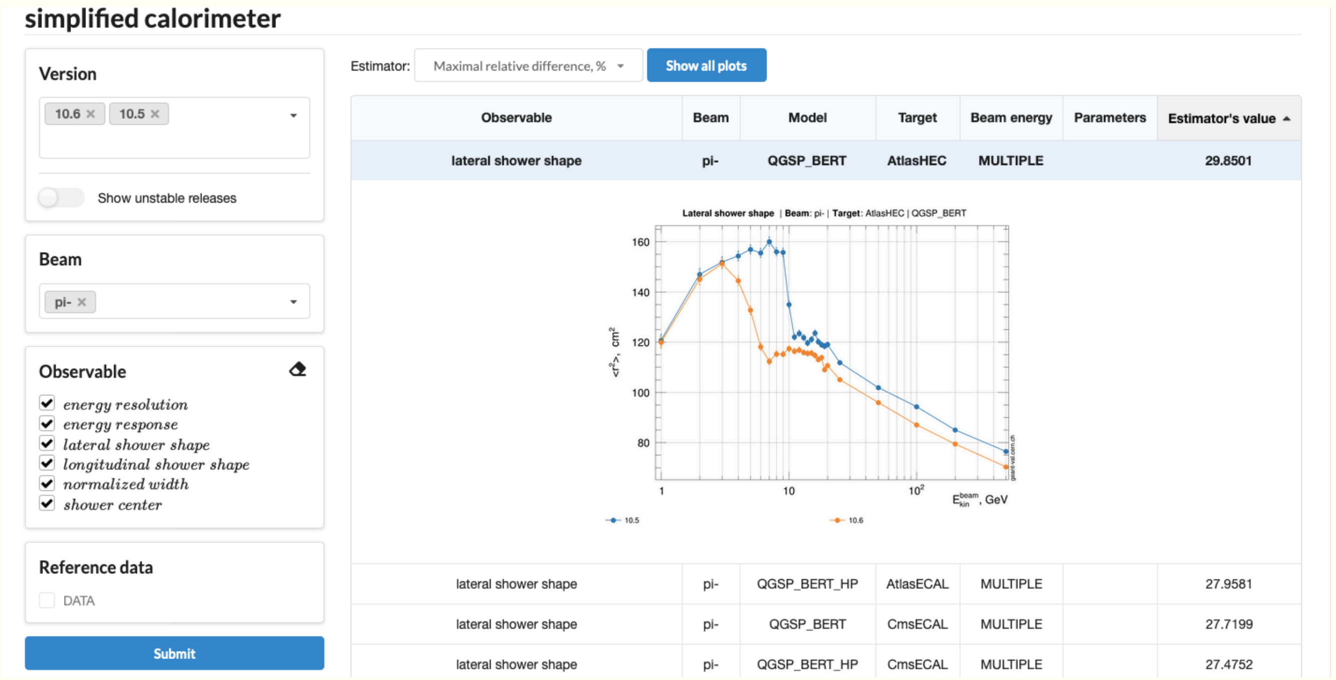
We are invited to present geant-val at FCC internal meeting.

Backup

- Backup slides

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.


simplified calorimeter

Version: 10.6 × 10.5 ×

Show unstable releases

Beam: pi- ×

Observable
- ✓ *energy resolution*
- ✓ *energy response*
- ✓ *lateral shower shape*
- ✓ *longitudinal shower shape*
- ✓ *normalized width*
- ✓ *shower center*

Reference data
- DATA

Submit

Estimator: Maximal relative difference, %

Show all plots

Observable	Beam	Model	Target	Beam energy	Parameters	Estimator's value
lateral shower shape	**pi-**	**QGSP_BERT**	**AtlasHEC**	**MULTIPLE**		**29.8501**
lateral shower shape	pi-	QGSP_BERT_HP	AtlasECAL	MULTIPLE		27.9581
lateral shower shape	pi-	QGSP_BERT	CmsECAL	MULTIPLE		27.7199
lateral shower shape	pi-	QGSP_BERT_HP	CmsECAL	MULTIPLE		27.4752

This is an experimental feature, a placeholder. Can be replaced by any estimator.
Results depends on what one stores as “errors” for histograms.

Validation workflow

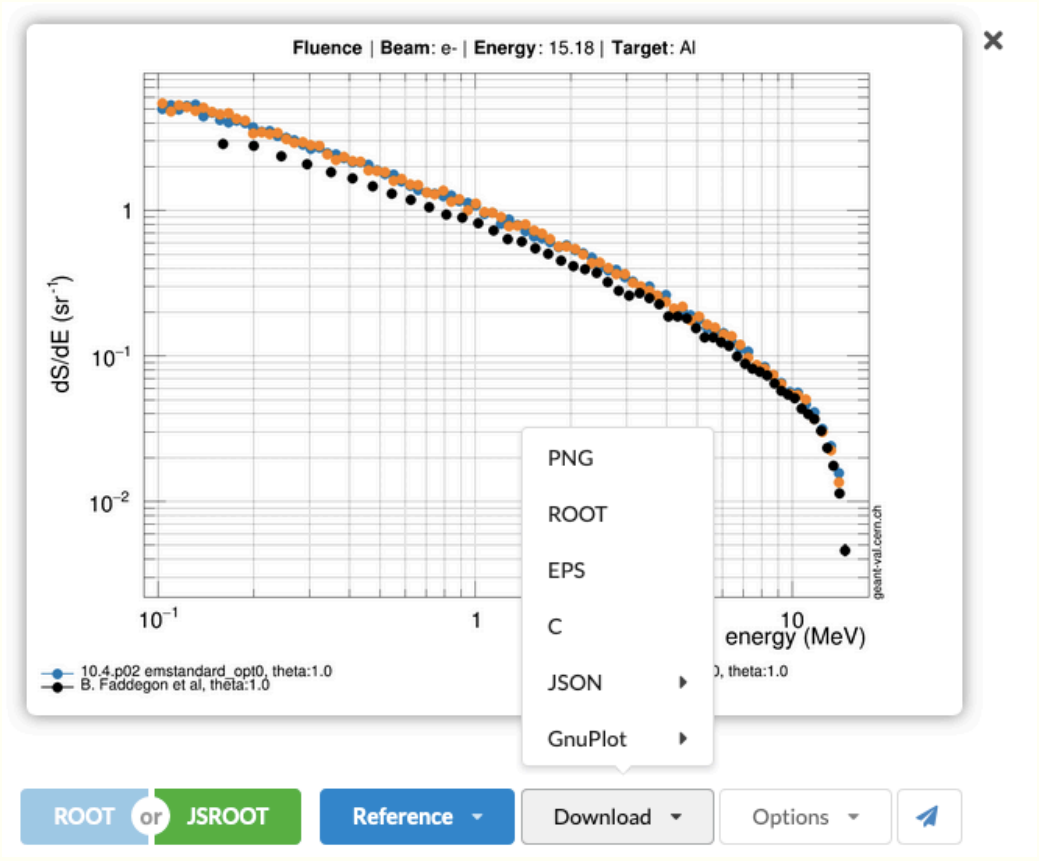
① Run test code locally, using batch systems or with GRID.

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

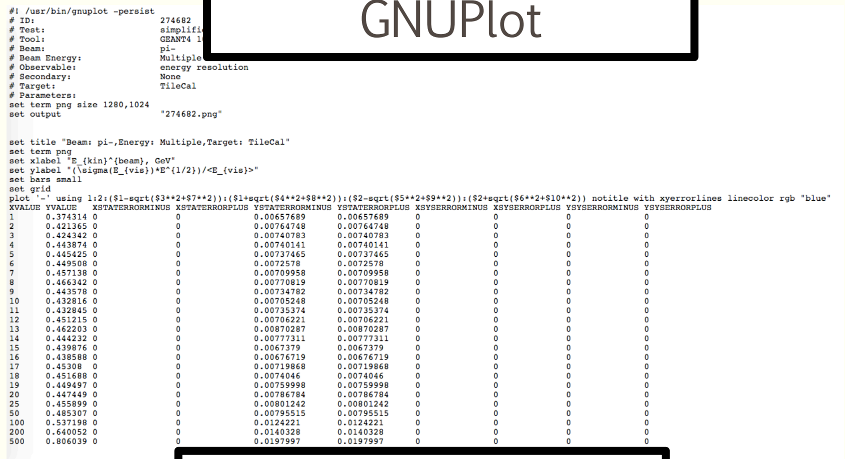
③ Upload JSONs using a dedicated python script into **geant-val**

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

data download formats

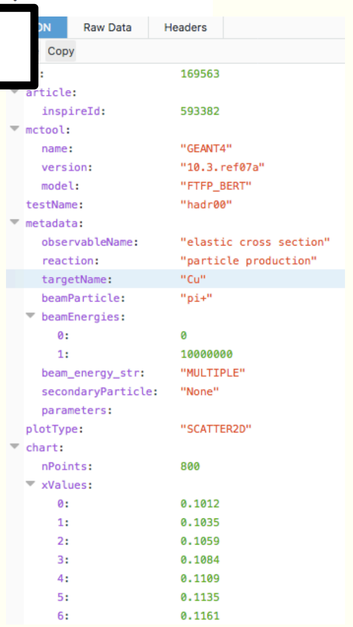


GNUPlot


```
#! /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
```

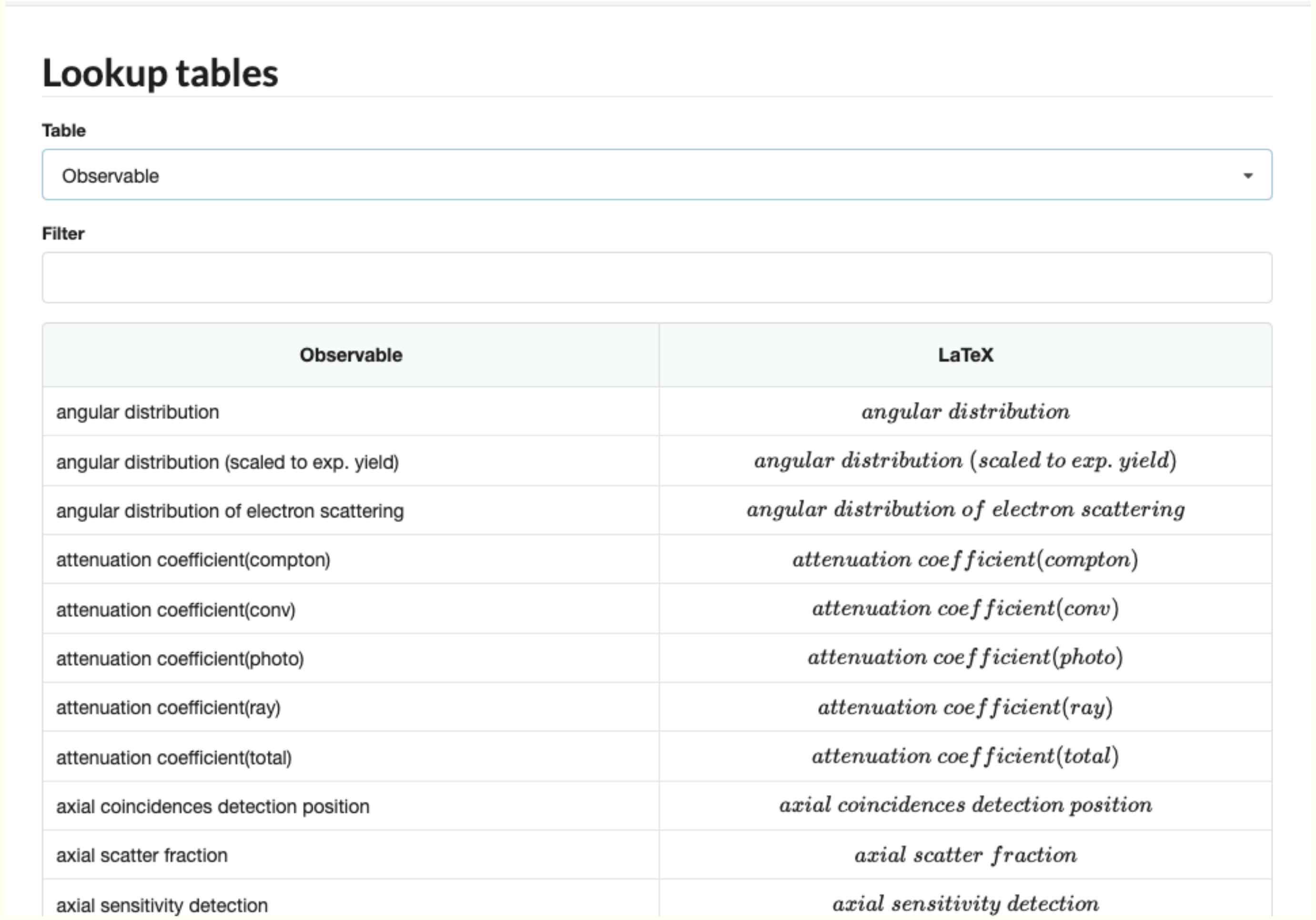
JSON format



```
:                       169563
article:
   inspireId:           593382
mctool:
   name:                "GEANT4"
   version:             "10.3.ref07a"
   model:               "FTFP_BERT"
testName:               "hadr00"
metadata:
   observableName:      "elastic cross section"
   reaction:            "particle production"
   targetName:          "Cu"
   beamParticle:        "pi+"
   beamEnergies:
      0:                0
      1:                10000000
   beam_energy_str:     "MULTIPLE"
   secondaryParticle:   "None"
   parameters:
plotType:               "SCATTER2D"
chart:
   nPoints:             800
   xValues:
      0:                0.1012
      1:                0.1035
      2:                0.1059
      3:                0.1084
      4:                0.1109
      5:                0.1135
      6:                0.1161
```

- JSON output = JSON input format

Validation table



Lookup tables

Table

Observable

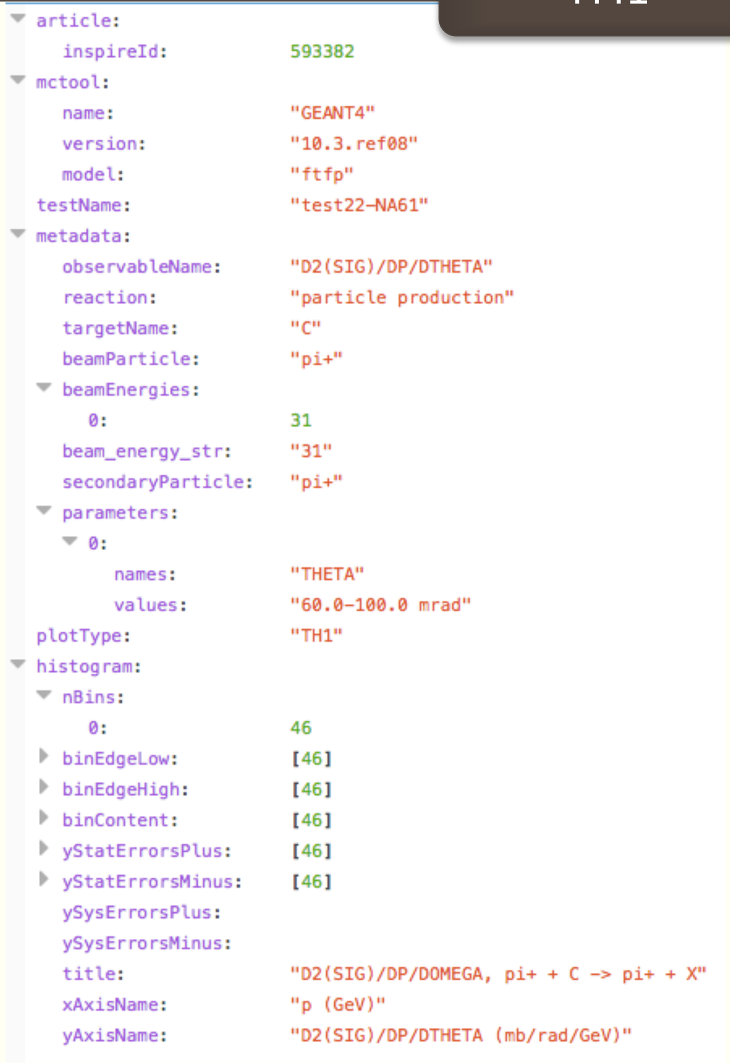
Filter

Observable	LaTeX
angular distribution	*angular distribution*
angular distribution (scaled to exp. yield)	*angular distribution (scaled to exp. yield)*
angular distribution of electron scattering	*angular distribution of electron scattering*
attenuation coefficient(compton)	*attenuation coef ficient(compton)*
attenuation coefficient(conv)	*attenuation coef ficient(conv)*
attenuation coefficient(photo)	*attenuation coef ficient(photo)*
attenuation coefficient(ray)	*attenuation coef ficient(ray)*
attenuation coefficient(total)	*attenuation coef ficient(total)*
axial coincidences detection position	*axial coincidences detection position*
axial scatter fraction	*axial scatter fraction*
axial sensitivity detection	*axial sensitivity detection*

Input JSON format

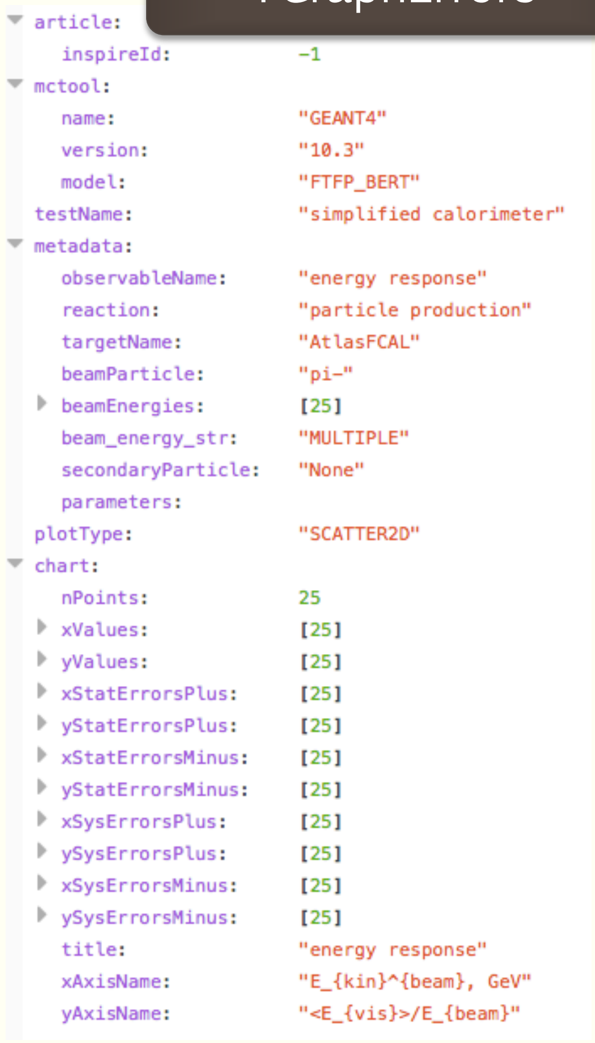
- Simple representation of histogram and corresponding metadata describing configuration/conditions.
- Human readable
- NB: metadata are kept in DB validation tables implement data integrity.

TH1



```
▼ article:
    inspireId:            593382
▼ mctool:
    name:                 "GEANT4"
    version:              "10.3.ref08"
    model:                "ftfp"
  testName:               "test22-NA61"
▼ metadata:
    observableName:       "D2(SIG)/DP/DTHETA"
    reaction:             "particle production"
    targetName:           "C"
    beamParticle:         "pi+"
  ▼ beamEnergies:
      0:                  31
    beam_energy_str:      "31"
    secondaryParticle:    "pi+"
  ▼ parameters:
    ▼ 0:
        names:            "THETA"
        values:           "60.0-100.0 mrad"
  plotType:               "TH1"
▼ histogram:
  ▼ nBins:
      0:                  46
  ▶ binEdgeLow:           [46]
  ▶ binEdgeHigh:          [46]
  ▶ binContent:           [46]
  ▶ yStatErrorsPlus:      [46]
  ▶ yStatErrorsMinus:     [46]
    ySysErrorsPlus:
    ySysErrorsMinus:
    title:                "D2(SIG)/DP/DOMEGA, pi+ + C -> pi+ + X"
    xAxisName:            "p (GeV)"
    yAxisName:            "D2(SIG)/DP/DTHETA (mb/rad/GeV)"
```

TGraphErrors



```
▼ article:
    inspireId:            -1
▼ mctool:
    name:                 "GEANT4"
    version:              "10.3"
    model:                "FTFP_BERT"
  testName:               "simplified calorimeter"
▼ metadata:
    observableName:       "energy response"
    reaction:             "particle production"
    targetName:           "AtlasFCAL"
    beamParticle:         "pi-"
  ▶ beamEnergies:         [25]
    beam_energy_str:      "MULTIPLE"
    secondaryParticle:    "None"
    parameters:
  plotType:               "SCATTER2D"
▼ chart:
    nPoints:              25
  ▶ xValues:              [25]
  ▶ yValues:              [25]
  ▶ xStatErrorsPlus:      [25]
  ▶ yStatErrorsPlus:      [25]
  ▶ xStatErrorsMinus:     [25]
  ▶ yStatErrorsMinus:     [25]
  ▶ xSysErrorsPlus:       [25]
  ▶ ySysErrorsPlus:       [25]
  ▶ xSysErrorsMinus:      [25]
  ▶ ySysErrorsMinus:      [25]
    title:                "energy response"
    xAxisName:            "E_{kin}^{beam}, GeV"
    yAxisName:            "<E_{vis}>/E_{beam}"
```

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!)

First step towards statistical evaluation

test22-HARP produces ~700 plots per Geant4 release

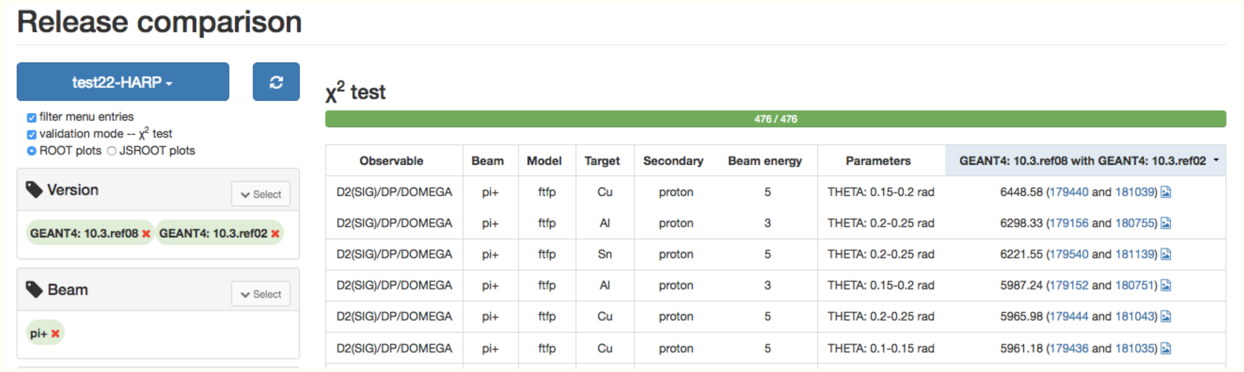
How to compare them with previous release of with experimental data?

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


Release comparison

χ^2 test

476 / 476

Observable	Beam	Model	Target	Secondary	Beam energy	Parameters	GEANT4: 10.3.ref08 with GEANT4: 10.3.ref02
D2(SIG)/DP/DOMEGA	pi+	ftfp	Cu	proton	5	THETA: 0.15-0.2 rad	6448.58 (179440 and 181039)
D2(SIG)/DP/DOMEGA	pi+	ftfp	Al	proton	3	THETA: 0.2-0.25 rad	6298.33 (179156 and 180755)
D2(SIG)/DP/DOMEGA	pi+	ftfp	Sn	proton	5	THETA: 0.2-0.25 rad	6221.55 (179540 and 181139)
D2(SIG)/DP/DOMEGA	pi+	ftfp	Al	proton	3	THETA: 0.15-0.2 rad	5987.24 (179152 and 180751)
D2(SIG)/DP/DOMEGA	pi+	ftfp	Cu	proton	5	THETA: 0.2-0.25 rad	5965.98 (179444 and 181043)
D2(SIG)/DP/DOMEGA	pi+	ftfp	Cu	proton	5	THETA: 0.1-0.15 rad	5961.18 (179436 and 181035)

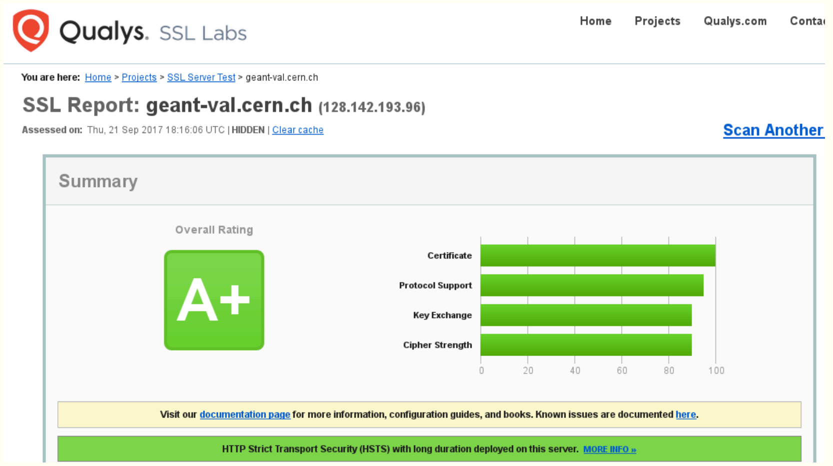
Security

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



You are here: Home > Projects > SSL Server Test > geant-val.cern.ch

SSL Report: geant-val.cern.ch (128.142.193.96)

Assessed on: Thu, 21 Sep 2017 18:16:06 UTC | HIDDEN | Clear cache

Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

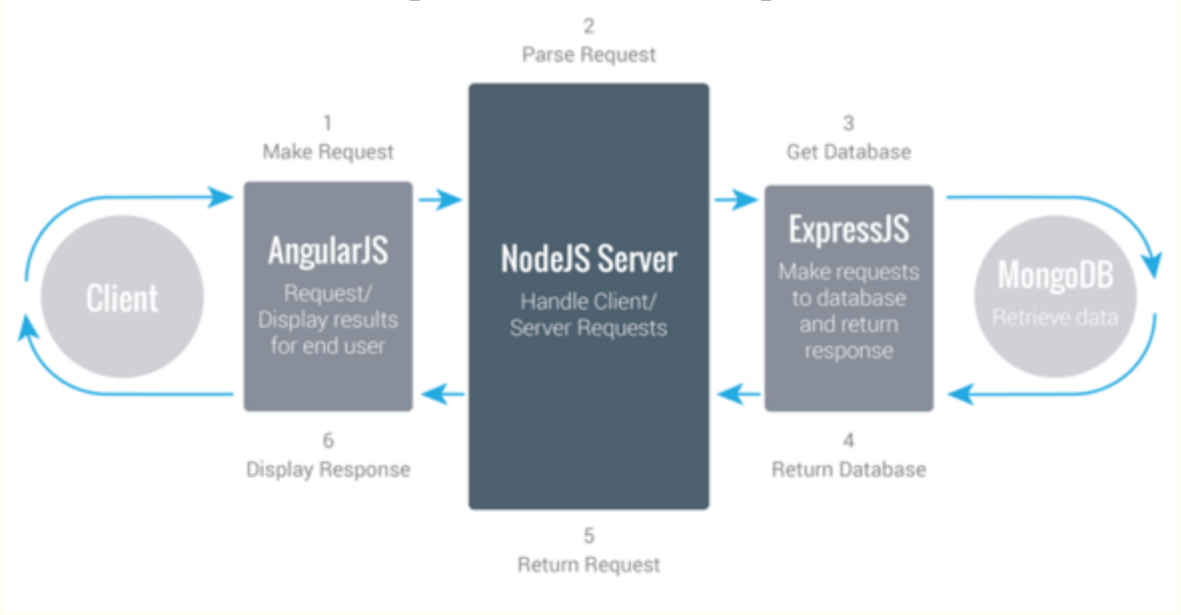
HTTP Strict Transport Security (HSTS) with long duration deployed on this server.

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.

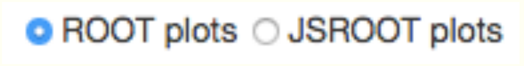
Webserver Architecture

- **Node.js** is an open-source, cross-platform JavaScript run-time environment for executing JavaScript code server-side
- Angular2 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



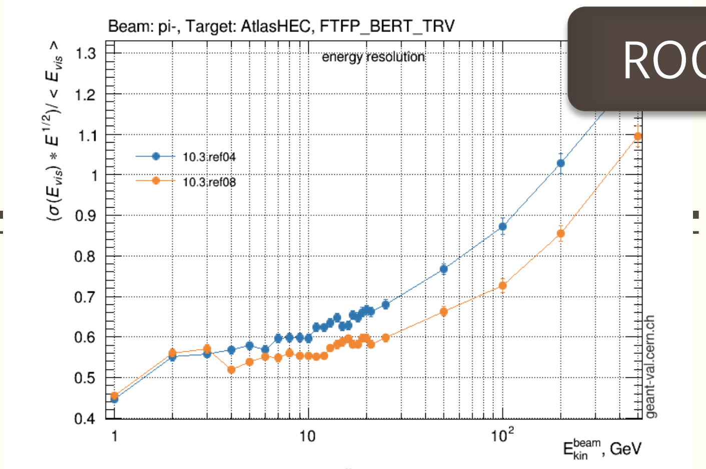
✓ Scalability
✓ Short development cycles
✓ Performance

Graphics/Plots

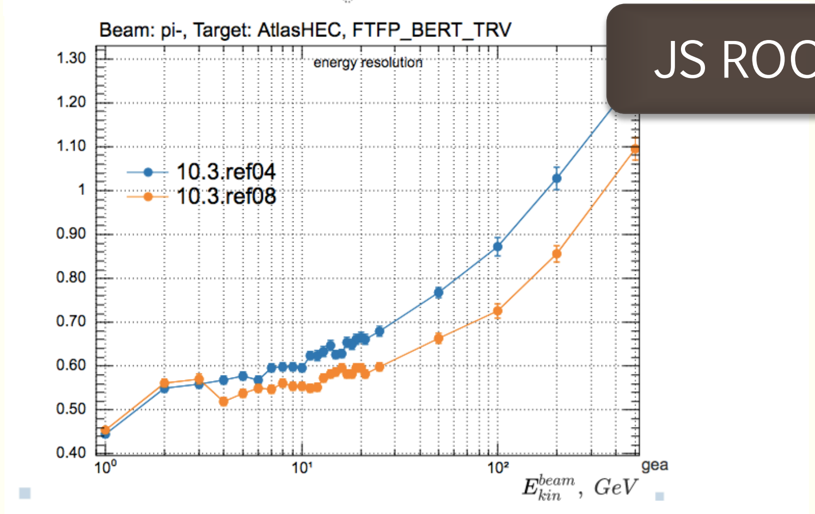


Since last Geant4 collaboration meeting we:

- added static ROOT plots – very slow
  - 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