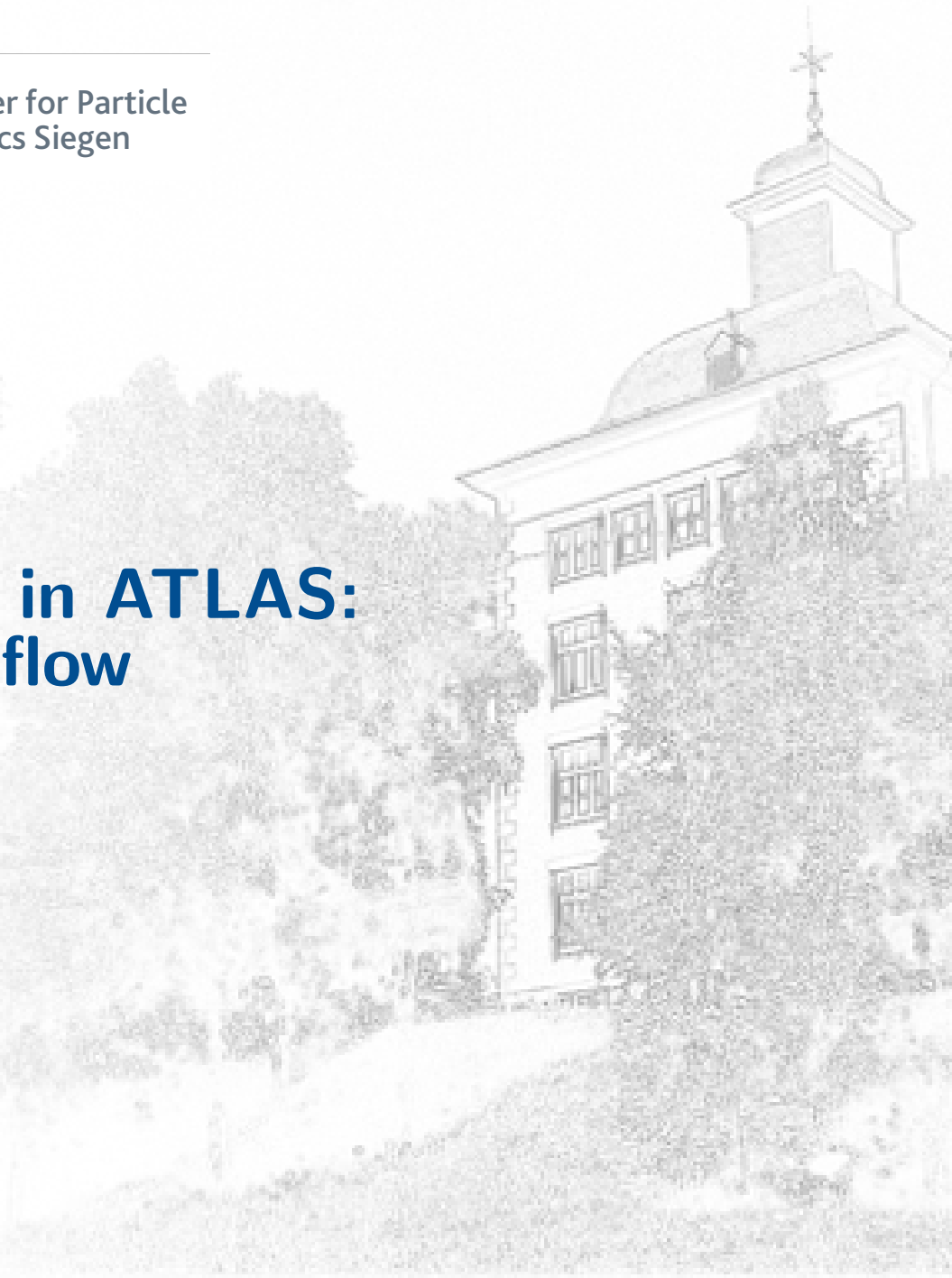
Universität Siegen

CPPS Center for Particle Physics Siegen

Katharina Voß

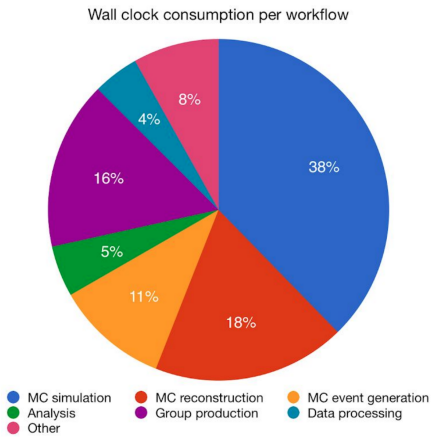
Monte-Carlo Generation in ATLAS: An overview of the workflow

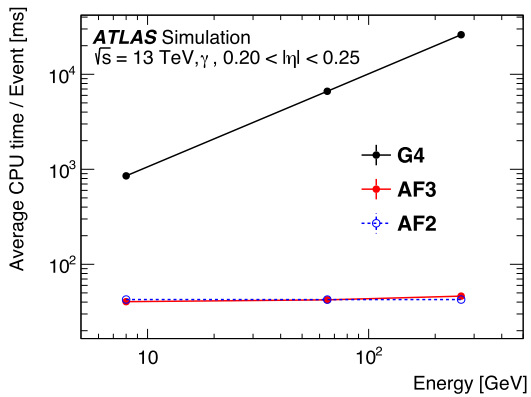


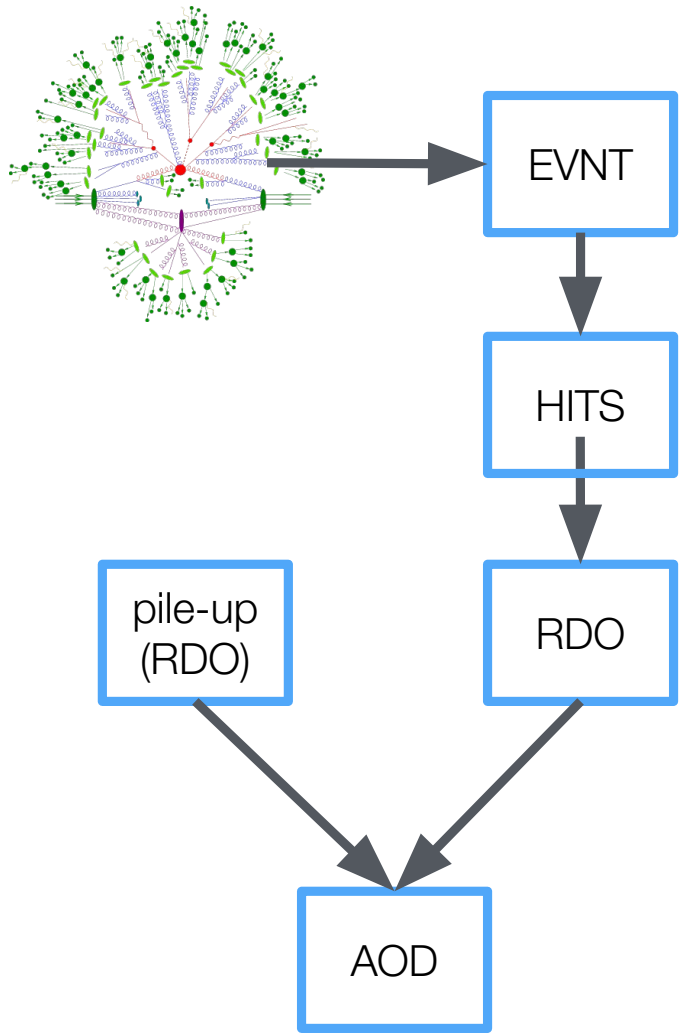
CPPS Retreat, Meinerzhagen

15th February 2024

Why is the generation of new samples slow?







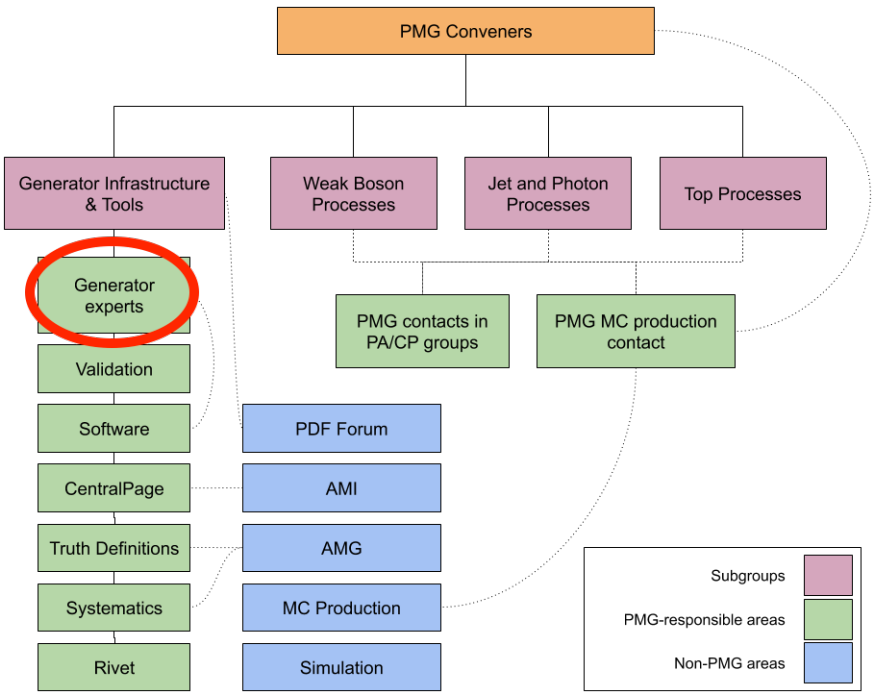
[CERN-LHCC-2020-015] [ComputSoftwBigSci6,7(2022)]

→ trying to give an overview / a feeling why the approval and the production of new Monte Carlo samples can take significant time in ATLAS

Generator software teams in ATLAS

Tasks of dedicated generator software expert teams within ATLAS

- ATLAS software maintenance and update for specific generator
- answering to technical and physics questions related to the generator



Generator
Powheg
MadGraph5_aMC@NLO
Sherpa
Pythia8
Herwig7

Uncertainty prescription for the $t\bar{t}$ process

For evaluation of the uncertainties marked in red new samples have to be produced

Hard process generation

- **scale uncertainty**
  varying $(K_R, K_F) \in \{(1, 0.5), (1, 2), (0.5, 1), (2, 1)\}$ with $\mu_{R/F} = K_{R/F}\mu_0$
- **PDF uncertainty**
  PDF4LHC variations added in quadrature (nominal NNPDF3.0)
- **NNLO reweighting**
  use NNLO reweighting based on truth $p_{\mathrm{T}}^{t\bar{t}}$, $m_{t\bar{t}}$, p_{T}^{t} and $p_{\mathrm{T}}^{\bar{t}}$
- **h_{damp} variation**
  compare nominal $h_{\mathrm{damp}} = 1.5 m_{\mathrm{top}}$ to $h_{\mathrm{damp}} = 3 m_{\mathrm{top}}$
- **top quark mass**
  compare nominal $m_{\mathrm{top}} = 172.5$ GeV to $m_{\mathrm{top}} \in \{172.0, 173.0\}$ GeV samples

Matching uncertainty

- **Pythia 8 $p_{\mathrm{T}}^{\mathrm{hard}}$ variation**
  compare nominal $p_{\mathrm{T}}^{\mathrm{hard}} = 0$ to $p_{\mathrm{T}}^{\mathrm{hard}} = 1$ sample

Uncertainty prescription for the $t\bar{t}$ process

Parton shower uncertainties

- **initial state radiation**
  Pythia 8 Var3c variations, vary ISR $\alpha_S(M_Z) = \{0.115, 0.140\}$, nominal $\alpha_S(M_Z) = 0.127$ [ATL-PHYS-PUB-2014-021]
- **final state radiation**
  vary μ_R in FSR splitting kernel by factor 0.5 and 2
- **parton shower**
  comparison of nominal Pwg+Py8 with Pwg+Herwig 7 (H7)

Specialised uncertainty definitions

- **top line shape** (only for analysis in off-shell region)
  comparison of nominal Pwg+Py8 to Pwg+MadSpin+Py8 $t\bar{t}$ sample
- **recoil-to-top**
  compare nominal (recoil-to-colour) to alternative recoil-to-top sample
- **underlying event**
  Py8 A14 Var1 variations (MPI and UE tuning parameters) [ATL-PHYS-PUB-2014-021]
- **colour reconnection**
  comparison of nominal to tuned results with different CR models CR1 and CR2 [ATL-PHYS-PUB-2017-008]

Monte Carlo statistics for $t\bar{t}$ in dileptonic decay

sample	statistics	simulation type
nominal (Pwhg+Py8)	550 M	FullSim
nominal (Pwhg+Py8)	550 M	FastSim
$h_{\text{damp}} = 3m_t$ (Pwhg+Py8)	310 M	FastSim
m_t variations (Pwhg+Py8)	2 x 230M	FastSim
$p_T^{\text{hard}} = 1$ (Pwhg+Py8)	310 M	FastSim
PS uncertaintiy (Pwhg+H7)	310 M	FastSim
top line shape (Pwhg+Py8)	310M	FastSim
recoil-to-top (Pwhg+Py8)	260 M	FastSim
UE variation (Pwhg+Py8)	2 x 90M	FastSim
colour reconnection (Pwhg+Py8)	2 x 190 M	FastSim

$\rightarrow$ if misconfiguration of the generator is found in the nominal sample, also systematic varied samples have to reproduced to not overestimate the modelling uncertainties
$\rightarrow$ careful validation is needed before nominal sample production can start

Outline

- integration of a new generator into ATLAS software
- generator software validation in ATLAS (external and internal software)
- Monte Carlo user request workflow

Integrating a new generator into ATLAS software

Release	e-tag	LCG layer	HepMC	Herwig7	MadGraph	Pythia8	Sherpa	EvtGen	Lhapdf	Starlight	Superchic	Rivet	Powheg ATLAS
22.6.0	e8328	98_ATLAS_6	2.6.11	7.2.1	2.8.1.atlas3	8.244	2.2.11	1.7.0	6.2.3	r313	3.0.6	3.1.2	04-05-01
22.6.1	e8337	98_ATLAS_6	2.6.11	7.2.1	2.8.1.atlas3	8.244	2.2.11	1.7.0	6.2.3	r313	3.0.6	3.1.2	04-05-01
22.6.2	e8345	98_ATLAS_8	2.6.11	7.2.1	2.9.2.atlas	8.244	2.2.11	1.7.0	6.2.3	r313	3.0.6	3.1.2	04-05-01
22.6.3	e8358	100_ATLAS_3	2.6.11	7.2.1	2.9.3.atlas1	8.304	2.2.11	1.7.0	6.2.3	r313	3.0.6	3.1.4	04-05-01

- compiled generator executable needs to be made available in software layer
- user configuration of the generator run is done in python job options scripts
- steering scripts to pass control from athena to the generator
- conversion scripts of the generated event into HepMC format
- storing of the events in root files or preparation for further downstream processing (e.g. simulation)

Integrating a new generator into ATLAS software

Release	e-tag	LCG layer	HepMC	Herwig7	MadGraph	Pythia8	Sherpa	EvtGen	Lhapdf	Starlight	Superchic	Rivet	Powheg ATLAS
22.6.0	e8328	98_ATLAS_6	2.6.11	7.2.1	2.8.1.atlas3	8.244	2.2.11	1.7.0	6.2.3	r313	3.0.6	3.1.2	04-05-01
22.6.1	e8337	98_ATLAS_6	2.6.11	7.2.1	2.8.1.atlas3	8.244	2.2.11	1.7.0	6.2.3	r313	3.0.6	3.1.2	04-05-01
22.6.2	e8345	98_ATLAS_8	2.6.11	7.2.1	2.9.2.atlas	8.244	2.2.11	1.7.0	6.2.3	r313	3.0.6	3.1.2	04-05-01
22.6.3	e8358	100_ATLAS_3	2.6.11	7.2.1	2.9.3.atlas1	8.304	2.2.11	1.7.0	6.2.3	r313	3.0.6	3.1.4	04-05-01

- compiled generator executable needs to be made available in software layer
- user configuration of the generator run is done in python job options scripts
- steering scripts to pass control from athena to the generator
- conversion scripts of the generated event into HepMC format
- storing of the events in root files or preparation for further downstream processing (e.g. simulation)

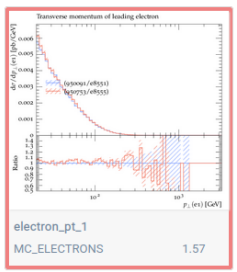
$\rightarrow$ compatible output format for downstream processing in ATLAS + setting up the event generation through python scripts for the end user is very convenient

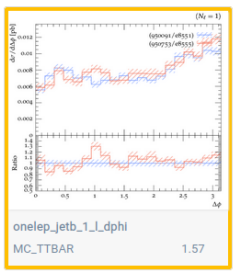
```
asetup AthGeneration,23.6.22
Gen_tf.py --ecmEnergy=13000.0 --jobConfig=410472
--outputEVNTFile=ttbar.EVNT.root --maxEvents=10000 --rivetAnas=MC_TTBAR
```

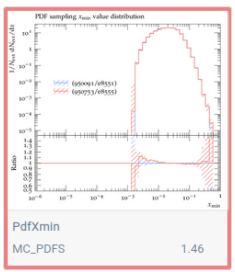
Implementation of new generators is done by software experts, as well as the implementation of new processes in an existing generator by the specific generator software experts in ATLAS (example: adding a new process in PowhegControl [!66549])

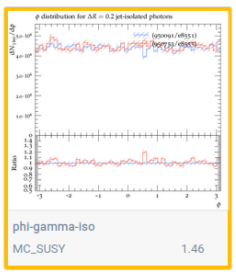
Generator software validation in ATLAS

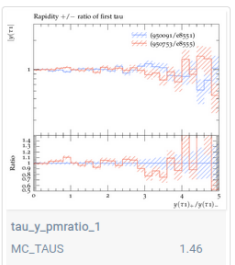
→ central validation of new generator versions and ATLAS software changes by comparing Rivet histograms [ATL-PHYS-PUB-2016-001] [JEM/PAVER]
→ dedicated MC validation team with biweekly reports in ATLAS sub group meeting (Generator, Infrastructure and Tools subgroup of Physics Modelling Group in ATLAS)

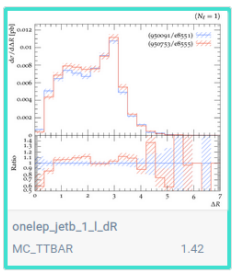

electron_pt_1
MC_ELECTRONS 1.57

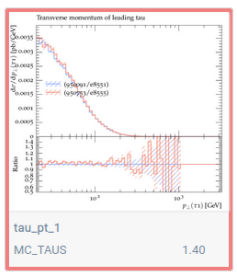

onelep_jetb_1_l_dphi
MC_TTBAR 1.57

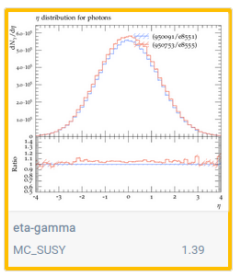

PdfXmin
MC_PDFS 1.46

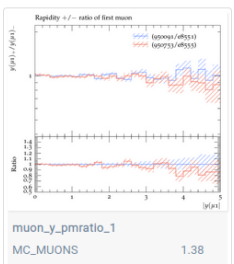

phi-gamma-iso
MC_SUSY 1.46

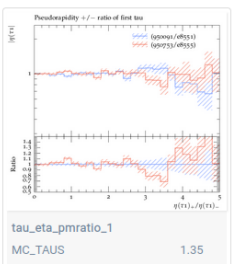

tau_y_pmratio_1
MC_TAUS 1.46

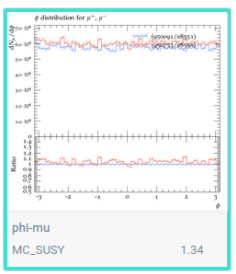

onelep_jetb_1_l_dR
MC_TTBAR 1.42

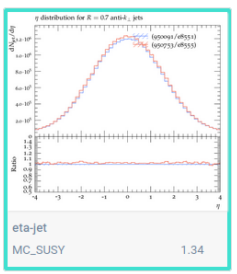

tau_pt_1
MC_TAUS 1.40


eta-gamma
MC_SUSY 1.39


muon_y_pmratio_1
MC_MUONS 1.38


tau_eta_pmratio_1
MC_TAUS 1.35


phi-mu
MC_SUSY 1.34


eta-jet
MC_SUSY 1.34

Requesting a new sample in ATLAS

Validation

- private production of EVNT samples (ME+PS+hadronisation) for validation
- depending on size of the request presentation in ATLAS working group meeting

Official Monte Carlo Request

- opening of an issue on JIRA (two persons per ATLAS WG are responsible for processing these requests)
- registration of the job options (python scripts for generator configuration and starting the generator run) in official git repository
  → several scripts checking e.g. the generated log files, job option names
  → in central production the job options are taken from this central git repository (not just documentation purpose)
- approval of the request by working group conveners
- approval of the request by Physics Modelling conveners
- submitting a production system request
  → information about the request is collected in Google spreadsheets, which are parsed to collect relevant information
- production system request is managed by production system team (monitoring, submitting simulation / reconstruction job after event generation is finished)

→ depending on the size of the request / occupation of the production system a MC request can take 2 weeks to 2 months after validation is finished

Conclusions and Outlook

Monte Carlo generation within ATLAS can be a long process

→ produced samples are used by multiple analyses
(not only by the analysis teams requesting the samples)

→ careful validation needed

→ responsibilities in ATLAS generator software and Monte Carlo request procedures clearly assigned

Which resources and skills are needed?

- ATLAS software environment
- C++ and Python knowledge (depends on specific task)
- Overview of current Monte Carlo studies

Conclusions and Outlook

Monte Carlo generation within ATLAS can be a long process

→ produced samples are used by multiple analyses
(not only by the analysis teams requesting the samples)

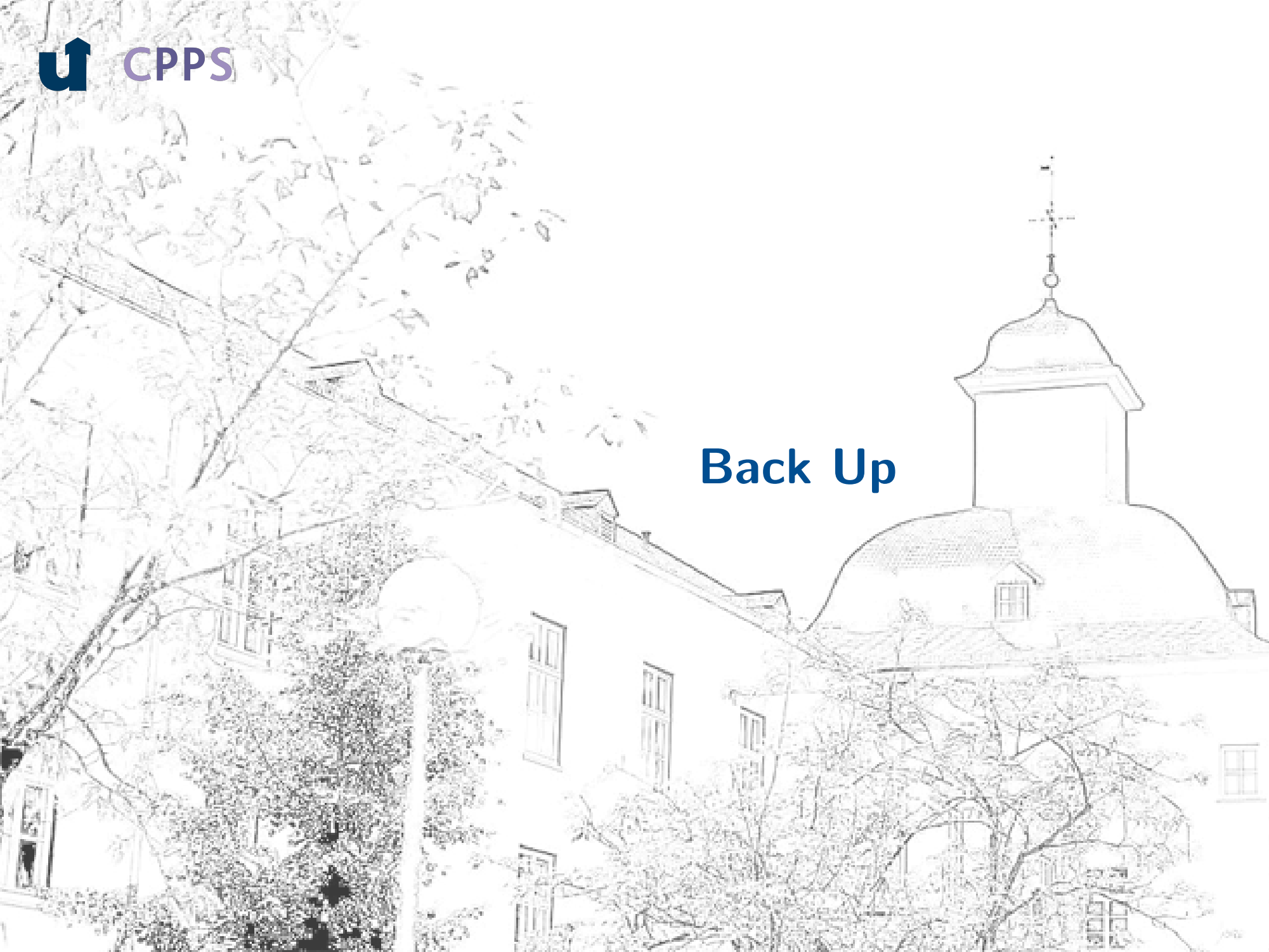
→ careful validation needed

→ responsibilities in ATLAS generator software and Monte Carlo request procedures clearly assigned

Which resources and skills are needed?

- ATLAS software environment
- C++ and Python knowledge (depends on specific task)
- Overview of current Monte Carlo studies

Thank you for your attention!



Back Up

Integrating a new generator into ATLAS software

Using the Powheg generator as an example

- **compiled generator executable needs to be made available in software layer**

Release	e-tag	LCG layer	HepMC	Herwig7	MadGraph	Pythia8	Sherpa	EvtGen	Lhapdf	Starlight	Superchic	Rivet	Powheg ATLAS
22.6.0	e8328	98_ATLAS_6	2.6.11	7.2.1	2.8.1.atlas3	8.244	2.2.11	1.7.0	6.2.3	r313	3.0.6	3.1.2	04-05-01
22.6.1	e8337	98_ATLAS_6	2.6.11	7.2.1	2.8.1.atlas3	8.244	2.2.11	1.7.0	6.2.3	r313	3.0.6	3.1.2	04-05-01
22.6.2	e8345	98_ATLAS_8	2.6.11	7.2.1	2.9.2.atlas	8.244	2.2.11	1.7.0	6.2.3	r313	3.0.6	3.1.2	04-05-01
22.6.3	e8358	100_ATLAS_3	2.6.11	7.2.1	2.9.3.atlas1	8.304	2.2.11	1.7.0	6.2.3	r313	3.0.6	3.1.4	04-05-01

- user configuration of the generator run is done in python job options scripts
- steering scripts to pass control from athena to the generator
- conversion scripts of the generated event into HepMC format
- storing of the events in root files or preparation for further downstream processing (e.g. simulation)

Integrating a new generator into ATLAS software

Using the Powheg generator as an example

- compiled generator executable needs to be made available in software layer
- **user configuration of the generator run needs to be configurable in python job options scripts**
  Example PowhegControl:
  - generating powheg.input card with ATLAS default values if not overwritten by the user in the job option

```
include('PowhegControl/PowhegControl_tt_Common.py')

PowhegConfig.decay_mode    = "t t~ > b l+ vl b~ l- vl~"

# Initial settings
PowhegConfig.hdamp         = 258.75                                    # 1.5 * mtop
```

  - scheduling needed pre- or postprocessing steps (e.g. Pythia PS)
- steering scripts to pass control from athena to the generator
- conversion scripts of the generated event into HepMC format
- storing of the events in root files or preparation for further downstream processing (e.g. simulation)

Integrating a new generator into ATLAS software

Using the Powheg generator as an example

- compiled generator executable needs to be made available in software layer
- user configuration of the generator run is done in python job options scripts
- **steering scripts to pass control from athena to the generator**
  Example PowhegControl:
  - powheg executable searched for and executed e.g. through [singlecore.py]

```
PowhegConfig.generate()
```

- conversion scripts of the generated event into HepMC format
- storing of the events in root files or preparation for further downstream processing (e.g. simulation)

Integrating a new generator into ATLAS software

Using the Powheg generator as an example

- compiled generator executable needs to be made available in software layer
- user configuration of the generator run is done in python job options scripts
- steering scripts to pass control from athena to the generator
- **conversion of the generated event into HepMC format**

```
//////////////////////////////////////////////////////////////////////////
StatusCode Pythia8_i::fillEvt(HepMC::GenEvent *evt){

  ATH_MSG_DEBUG(">>> Pythia8_i from fillEvt");

  evt->set_event_number(m_internal_event_number);

  // if using "getGroupWeight" and | lhastrategy | = 4, then need to convert mb to pb ( done otherwise when calling info.weight(), [...] )
  if( m_internal_event_number == 1 && std::abs(m_pythia->info.lhaStrategy()) == 4 ) {
     m_conversion = ( (double) PYTHIA8_CONVERSION);
     ATH_MSG_DEBUG(" LHA strategy needs a conversion to fix Pythia8 shower weights bug(s) equal to " << m_conversion);
  }


  if(m_pythia->event.size() < 2){
    ATH_MSG_ERROR("Something wrong with this event - it contains fewer than 2 particles!");
    ATH_MSG_ERROR("internal event number is "<<m_internal_event_number);
    return StatusCode::FAILURE;
  }

  m_pythiaToHepMC.fill_next_event(*m_pythia, evt, m_internal_event_number);

  if(m_lheFile != "" && m_storeLHE) addLHEToHepMC(evt);
```

- storing of the events in root files or preparation for further downstream processing (e.g. simulation)

Integrating a new generator into ATLAS software

Using the Powheg generator as an example

- compiled generator executable needs to be made available in software layer
- user configuration of the generator run is done in python job options scripts
- steering scripts to pass control from athena to the generator
- conversion scripts of the generated event into HepMC format
- **storing of the events in root files or preparation for further downstream processing (e.g. simulation)**
  $\rightarrow$ this is done centrally, no generator specific implementation is needed

Integrating a new generator into ATLAS software

Using the Powheg generator as an example

- compiled generator executable needs to be made available in software layer
- user configuration of the generator run is done in python job options scripts
- steering scripts to pass control from athena to the generator
- conversion scripts of the generated event into HepMC format
- storing of the events in root files or preparation for further downstream processing (e.g. simulation)

→ compatible output format for downstream processing in ATLAS + setting up the event generation through python scripts for the end user is very convenient

```
asetup AthGeneration,23.6.22
Gen_tf.py --ecmEnergy=13000.0 --jobConfig=410472
--outputEVNTFile=ttbar.EVNT.root --maxEvents=10000 --rivetAnas=MC_TTBAR
```

Implementation of new generators is done by software experts, as well as the implementation of new processes in an existing generator by the specific generator software experts in ATLAS (example: adding a new process in PowhegControl [!66549])

Currently dedicated generator expert teams in ATLAS for Powheg, Pythia, Herwig, Sherpa, MadGraph