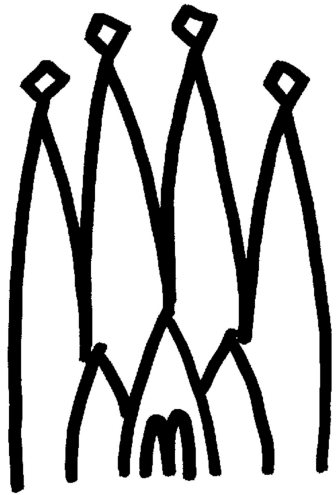
SAMPO

Gaudi-based framework status report

25 Feb 2026

Brief overview

- Based on Gaudi:
  - Intra- and Inter-Event parallelism
  - Configuring via python
  - Easy to develop new components (established architecture)
- Current functionality
  - Pythia8 for events generation (multithreaded)
  - GeoModel for detector description
  - HepMC3 as main format, HepMC2-compatible
  - Geant4 for simulation (multithreaded)
  - *JobConfigurator* for job configuring
- Links
  - gitlab: https://git.jinr.ru/spd/spd-sw/sampo/-/tree/develop?ref_type=heads
  - docker images: https://git.jinr.ru/spd/spd-sw/bisgm



Gaudi terminology

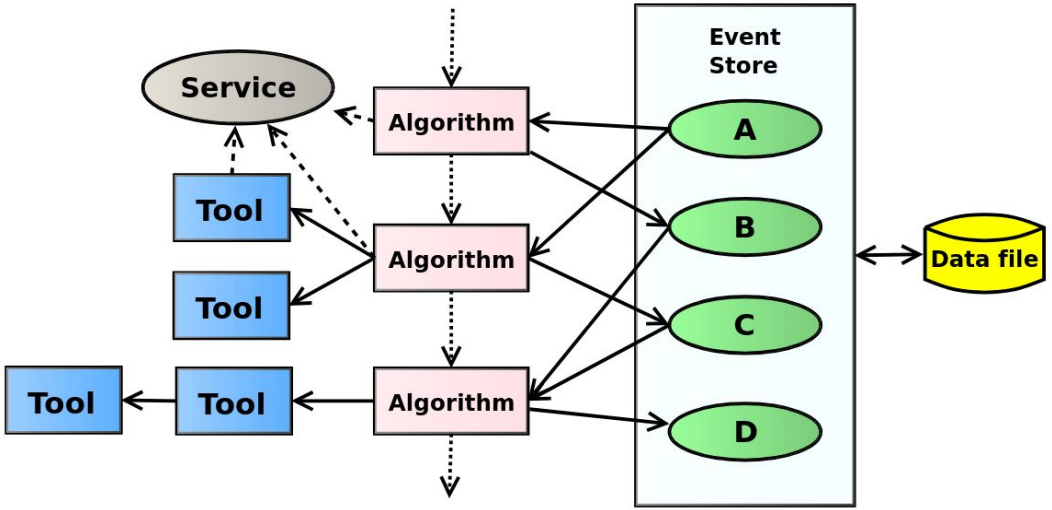
DataObjects - any quantity-like objects (hits, tracks, etc.)

Algorithms - anything that can be represented as procedure

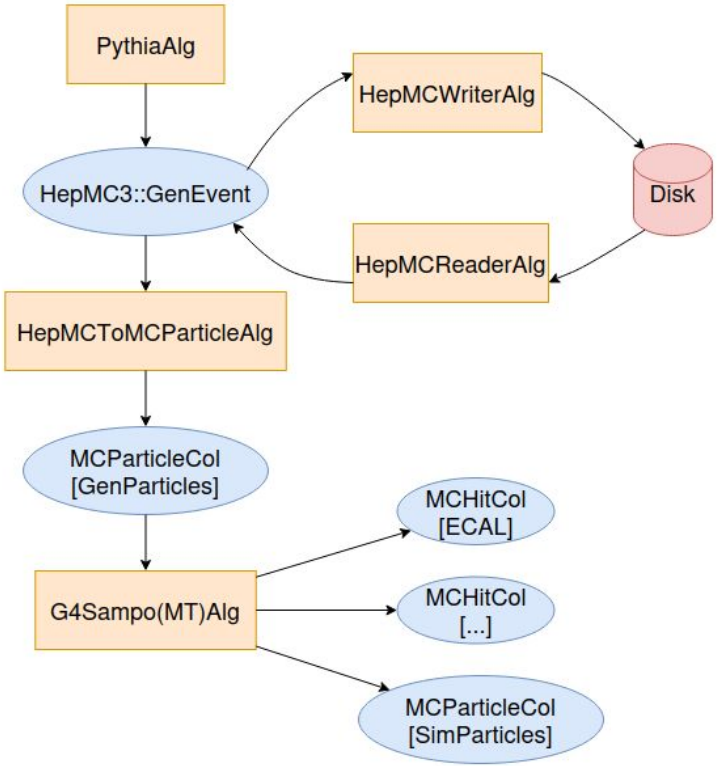
Services - global objects used by other components

Tools - encapsulated subprocedures that can be injected to other components

Transient event store (TES) - service which stores data objects and thus allows components to exchange data



Data flow in generation and simulation



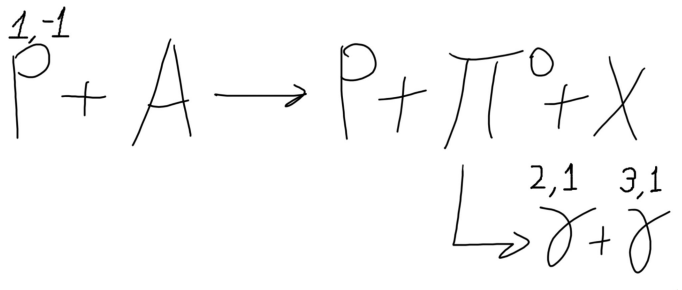
- HepMC3 is universal format for event generators
- HepMC3 is used to create MCParticles
- Geant4 Algorithm takes MCParticles as input and produces collections of universal MCHits for each subsystem. Also it produces collection of “interesting” MCParticles

“Interesting” Particles (SimParticles)

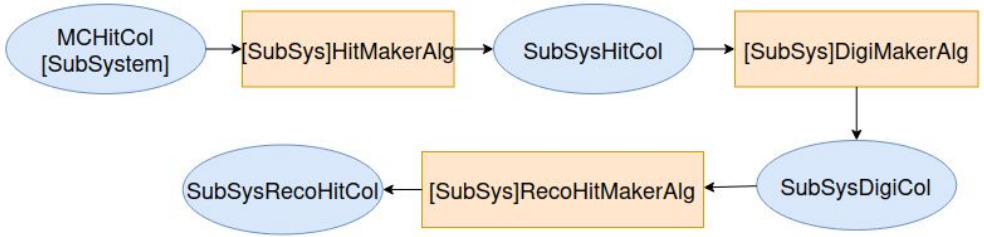
Track is considered to be interesting if:

1) it is primary
2) it is a product of decay process
3) it is product of gamma conversion

If track is selected as interesting then new MCParticle is created and it’s most recent “interesting” ancestor is marked as it’s parent. Parent particle id for primary particles is -1. Each MCHitskeeps id of “interesting” particle, which (or not “interesting” children of which) produced it.



Subsystem simulation data flow

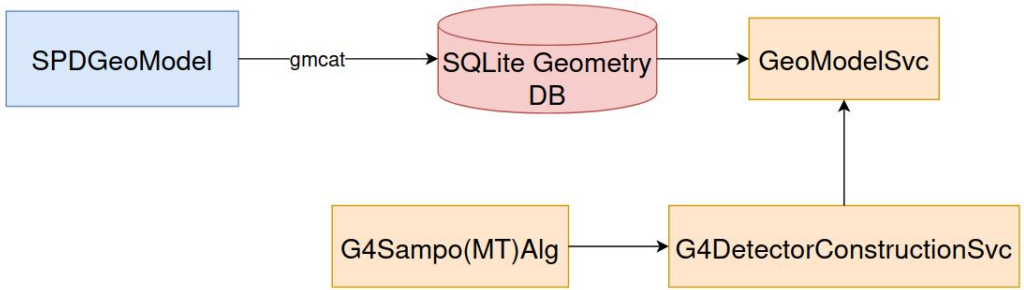


SybSysHit is individual for each subsystem and is constructed based on data stored in MCHits. For Straw Tracker case (just example) it can contain drift time and charge.

MCHits are produced by Geant4. MCHit aggregates information (energy deposition, global/local position, time, detector id, particle id) over single particle propagation step. Thus one particle can produce many MCHits in sensitive region of the detector.

SubSysDigi is produced from SubSysHits. Digis contain data which comes from real detector (smeared by the readout electronics and possible pile-ups).

Geometry Description

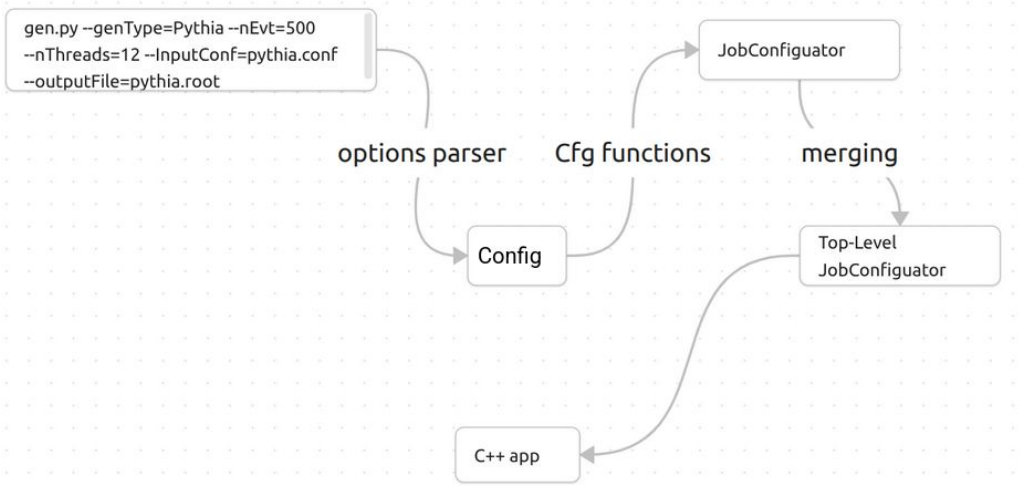


Simulation status

SubSystem	GeoModel description	Sampo Hit Algorithms
Micromegas Central Tracker	×	×
Double-Sided Silicon Vertex Detector	×	×
MAPS Vertex Detector	×	×
Straw Tracker	Barrel only	×
Inner Beam-Beam Counter	×	×
Outer Beam-Beam Counter	×	×
Range System	(See Alexandr's talk today)	×
Zero-Degree Calorimeter	✓	×
Electromagnetic Calorimeter	✓	✓
Time-Of-Flight System	×	×
FARICH	×	×

JobConfigurator

- Gaudi has native GaudiConfig2 configuring system but it is not convenient and has bugs
- Based on Athena's ComponentAccumulator (https://atlas-software.docs.cern.ch/athena/configuration/ca/) JobConfigurator was introduced to Sampo



JobConfigurator example

```python
from GaudiConfig2 import Configurables as C
from SampoConfiguration.JobConfigurator import JobConfigurator

def PythiaHelperSvcCfg(config):
    jc = JobConfigurator()
    pythiaHelperSvc = C.PythiaHelperSvc()
    pythiaHelperSvc.settingsFileName = config.EvGen.Pythia.ConfigPath
    pythiaHelperSvc.printBanner = config.EvGen.Pythia.PrintBanner
    pythiaHelperSvc.retryCountMax = config.EvGen.Pythia.RetryCountMax
    jc.addService(pythiaHelperSvc, primary=True)
    return jc

def PythiaAlgCfg(config):
    jc = JobConfigurator()
    pythiaAlg = C.PythiaAlg()
    pythiaAlg.PythiaHelperSvc = jc.popPrimaryAndMerge(PythiaHelperSvcCfg(config))
    nthreads = config.Concurrency.NThreads
    if nthreads > 0:
        nslots = config.Concurrency.NEventSlots
        if nslots > 0:
            pythiaAlg.Cardinality = nslots
        else:
            pythiaAlg.Cardinality = nthreads

    jc.addAlgorithm(pythiaAlg)
    return jc
```

```python
#! /usr/bin/env python

from SampoConfiguration.ConfigBuilderSetup import initConfigBuilder
from SampoConfiguration.MainServicesConfig import MainServicesCfg
from Generators.ConfigBuilderSetup import registerEvGenSettings

def main():
    configBuilder = initConfigBuilder()
    registerEvGenSettings(configBuilder)
    configBuilder.buildConfig()
    configBuilder.readCLI()
    config = configBuilder.getLockedConfig()

    jc = MainServicesCfg(config)

    if(config.EvGen.Type == 'Pythia'):
        from Pythia8.Pythia8Config import PythiaAlgCfg
        jc.merge(PythiaAlgCfg(config))
    else:
        raise RuntimeError(f"Unknown (or not set) EvGen.Type: {config.EvGen.Type}")

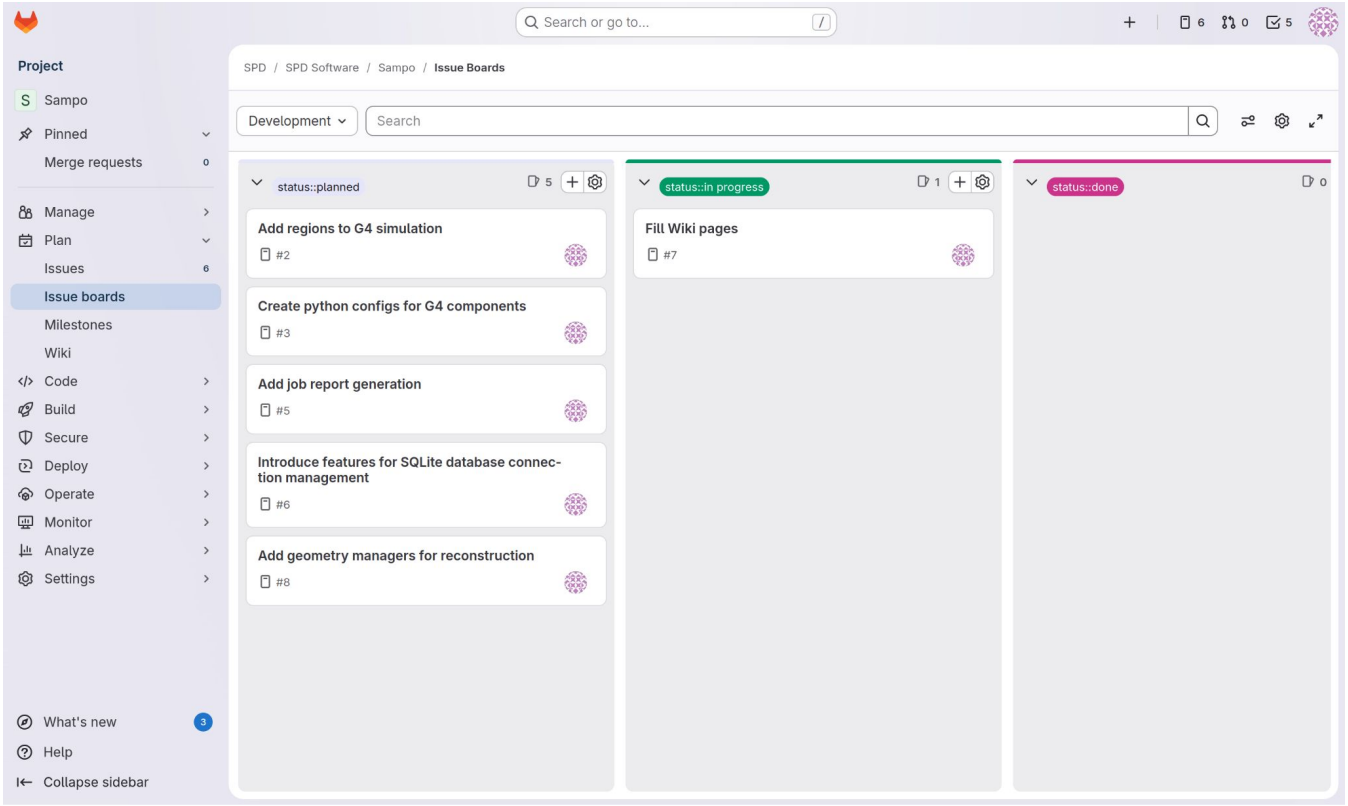
    if(config.EvGen.HepMC.OutputPath != ''):
        from HepMC.HepMCConfig import HepMCWriterAlgCfg
        jc.merge(HepMCWriterAlgCfg(config))
    jc.run()

if __name__ == '__main__':
    main()
```

Updates since SPD X Collaboration Meeting

- JobConfigurator was introduced
- Interesting particles arising during simulation are now saved and can be used by downstream algorithms
- Added HepMC2 compatibility
- Added G4ParticleGunAlg for debugging and testing purposes
- Added feature that allows to update materials exported from GeoModel
- Added possibility to add arbitrary amount of Event- and Tracking- actions for Geant4 simulation
- Added pytests

What is planned



What is required from the community

- Subsystem teams are sincerely asked to start contributing to both SPDGeoModel (detector description) and Sampo (Hit/Digi Algorithms) projects. For some of subsystems there are still no responsible persons assigned for these tasks. Straw-Tracker is our top priority.
- Feedback and feature requests would be appreciated
- We would be glad to welcome new members to the SAMPO core team