SAMPO

Geant4 integration status

SPD S&C meeting, February 18, 2025

Updates

Container registry: https://git.jinr.ru/spd/spd-sw/gaudi (check develop branch)

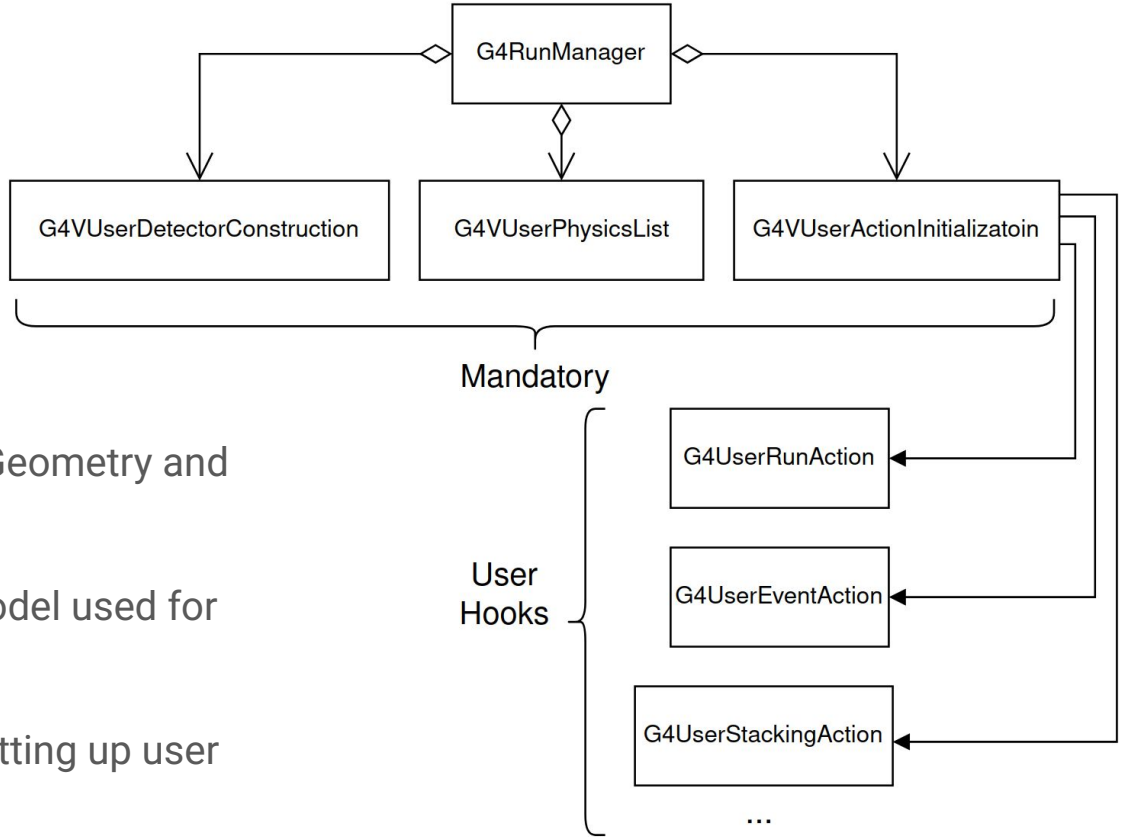
Pull Gaudi to try hands-on. Also available:

- Geant4
- GeoModel
- many other Gaudi dependencies

FINALLY!!!

Sampo repo: https://git.jinr.ru/spd/spd-sw/sampo (also develop branch)

Geant4: Features

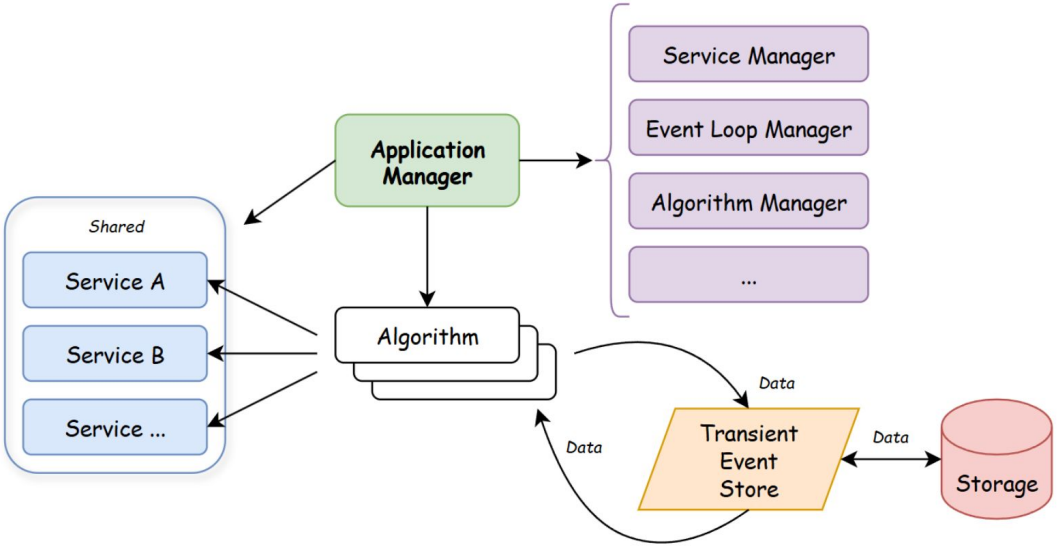


G4VUserDetectorConstruction -> Geometry and sensitive parts of the detector

G4VUserPhysicsList -> Physics model used for calculations

G4VUserActionInitioalization -> Setting up user hooks

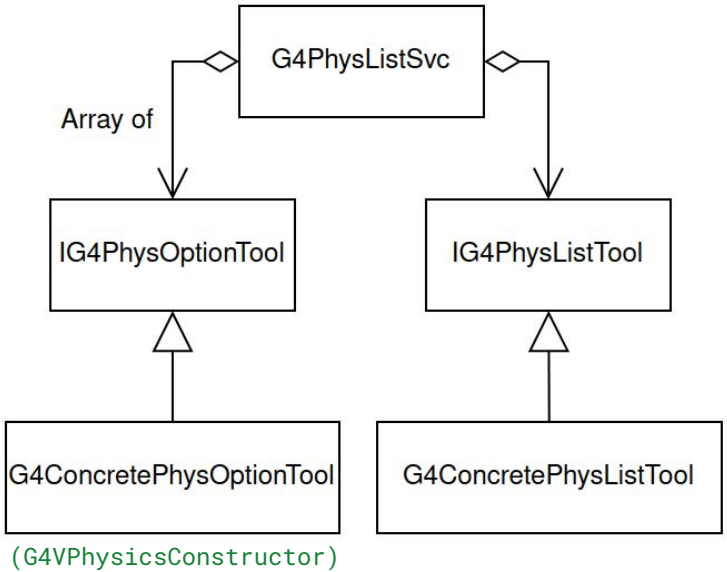
Gaudi: Architecture



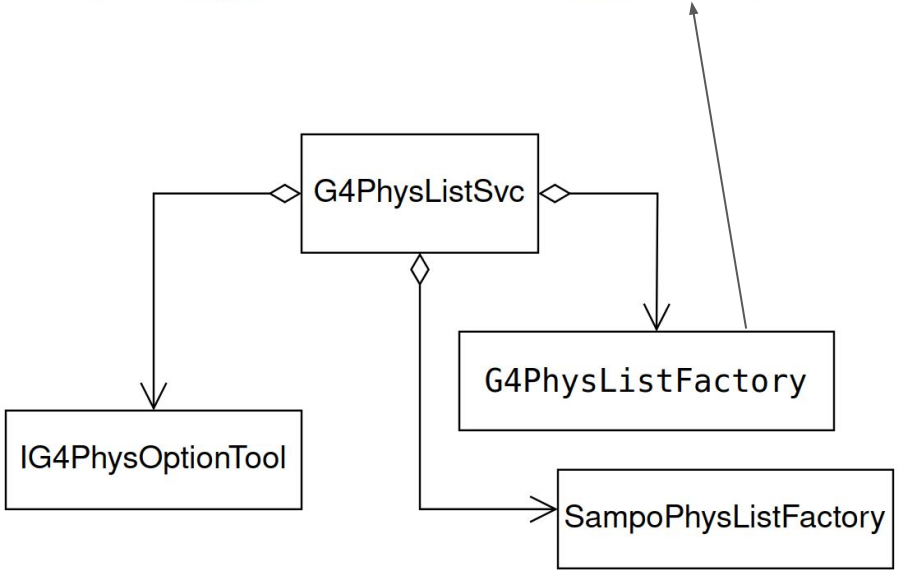

```python
from Configurables import ExampleAlg, ExampleTool

from Gaudi.Configuration import ApplicationMgr

evt_max = 10
evt_sel = "NONE"

my_algo = ExampleAlg("A1")
my_algo.propertyName = "cwebuciuwe"

tool = ExampleTool("ExampleTool1")
tool.String = "Is a private tool"
my_algo.PrivateTool = tool

tool2 = ExampleTool("ExampleTool2")
tool2.String = "Is a public tool"
my_algo.PublicTool = tool2

# create ApplicationMgr and start Gaudi app (C++)
ApplicationMgr(
    EvtMax=evt_max,
    EvtSel="NONE",
    TopAlg=[my_algo]
)
```

PhysicsList

```
G4VModularPhysicsList* p = nullptr;
if(had_name == "FTFP_BERT")              {p = new FTFP_BERT(verbose);}
else if(had_name == "FTFP_BERT_HP")      {p = new FTFP_BERT_HP(verbose);}
else if(had_name == "FTFP_BERT_TRV")     {p = new FTFP_BERT_TRV(verbose);}
else if(had_name == "FTFP_BERT_ATL")     {p = new FTFP_BERT_ATL(verbose);}
else if(had_name == "FTFQGSP_BERT")      {p = new FTFQGSP_BERT(verbose);}
else if(had_name == "FTFP_INCLXX")       {p = new FTFP_INCLXX(verbose);}
else if(had_name == "FTFP_INCLXX_HP")    {p = new FTFP_INCLXX_HP(verbose);}
else if(had_name == "FTF_BIC")           {p = new FTF_BIC(verbose);}
```

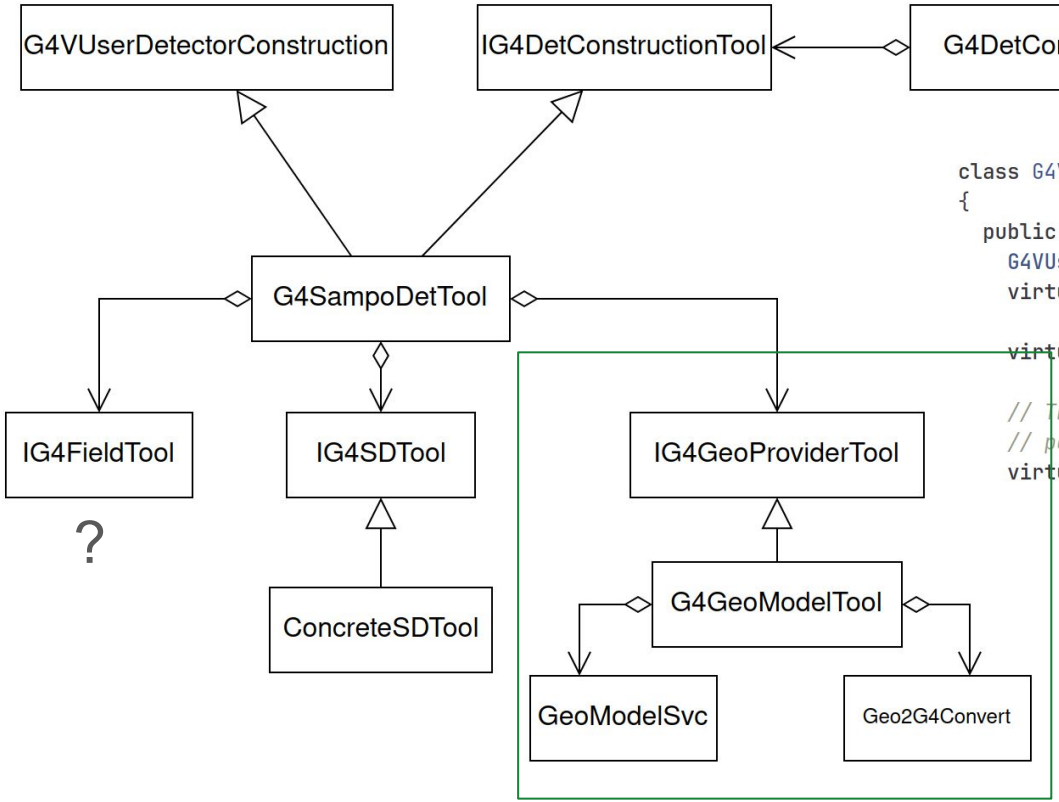


(G4VPhysicsConstructor)



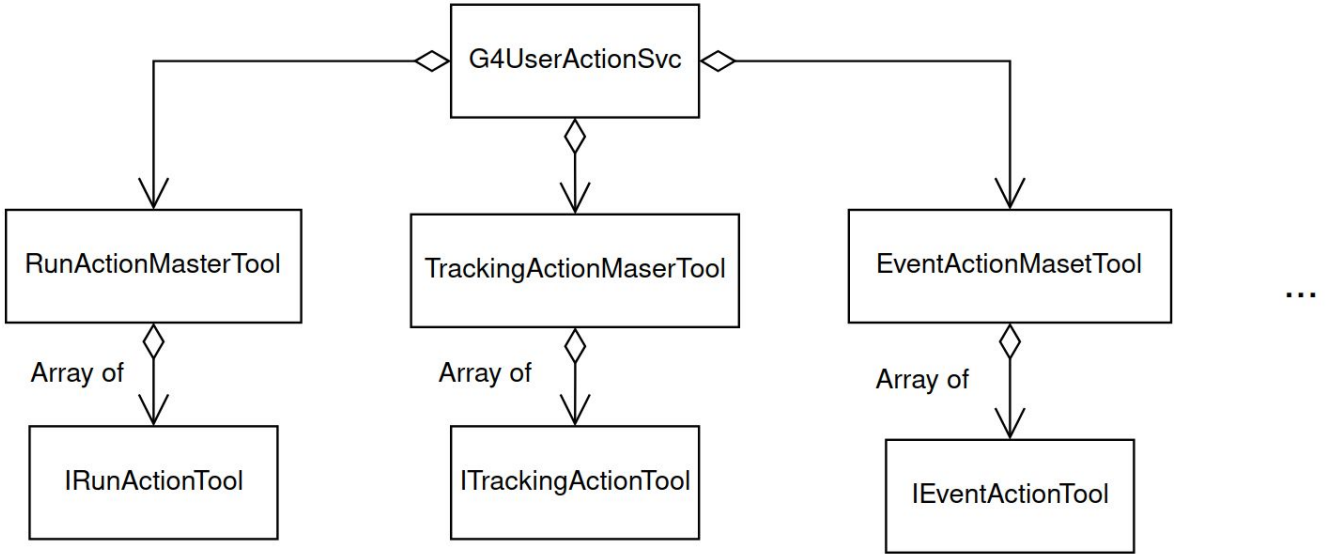

```
class myHooks : public UserHooks{
  ....
};

UserHooksFactory::Factory<myHooks> myHooksFactory("myHooks");
```

DetectorConstruction

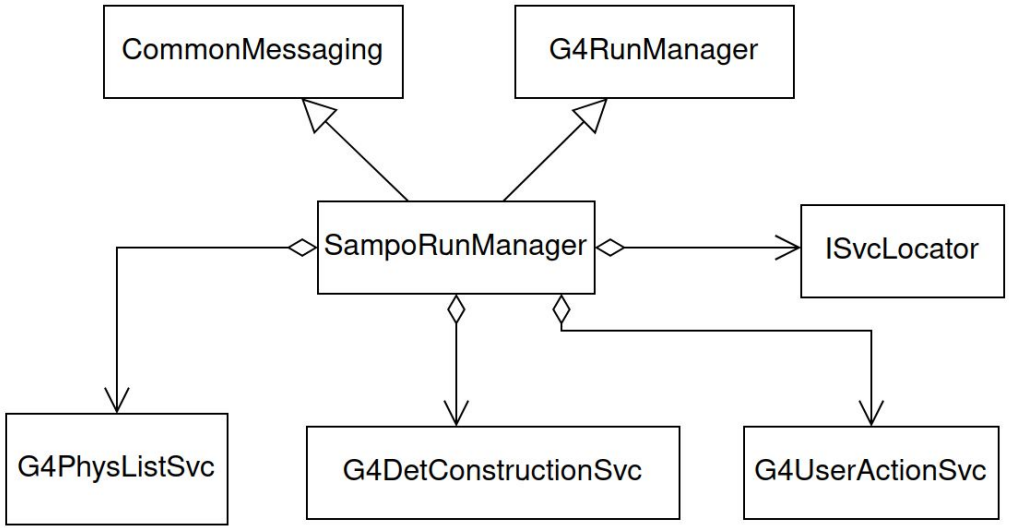



```
class G4VUserDetectorConstruction
{
  public:
    G4VUserDetectorConstruction() = default;
    virtual ~G4VUserDetectorConstruction() = default;

    virtual G4VPhysicalVolume* Construct() = 0;

    // This method is used in multi-threaded applications to build
    // per-worker non-shared objects: SensitiveDetectors and Field managers.
    virtual void ConstructSDandField();
```

UserActions



RunManager




```
class G4AtlasRunManager: public G4RunManager, public AthMessaging {

public:

  virtual ~G4AtlasRunManager() {}

  /// Retrieve the singleton instance
  static G4AtlasRunManager* GetG4AtlasRunManager ATLAS_NOT_THREAD_SAFE ();

  /// Does the work of simulating an ATLAS event
  bool ProcessEvent(G4Event* event);

  /// G4 function called at end of run
  void RunTermination() override final;

  /// Configure the detector geometry service handle
  void SetDetGeoSvc(const std::string& typeAndName) {
    m_detGeoSvc.setTypeAndName(typeAndName);
  }

  /// Configure the Fast Simulation Master Tool handle
  void SetFastSimMasterTool(const std::string& typeAndName) {
    m_fastSimTool.setTypeAndName(typeAndName);
  }

  /// Configure the Physics List Tool handle
  void SetPhysListSvc(const std::string& typeAndName) {
    m_physListSvc.setTypeAndName(typeAndName);
  }
```

```
// --------------------------------------------------------------------
void G4RunManager::BeamOn(G4int n_event, const char* macroFile, G4int n_select)
{
  fakeRun = n_event <= 0;
  G4bool cond = ConfirmBeamOnCondition();
  if (cond) {
    numberOfEventToBeProcessed = n_event;
    numberOfEventProcessed = 0;
    ConstructScoringWorlds();
    RunInitialization();
    DoEventLoop(n_event, macroFile, n_select);
    RunTermination();
  }
  fakeRun = false;
}
```

```
// --------------------------------------------------------------------
void G4RunManager::DoEventLoop(G4int n_event, const char* macroFile, G4int n_select)
{
  InitializeEventLoop(n_event, macroFile, n_select);

  // Event loop
  for (G4int i_event = 0; i_event < n_event; ++i_event) {
    ProcessOneEvent(i_event);
    TerminateOneEvent();
    if (runAborted) break;
  }

  // For G4MTRunManager, TerminateEventLoop() is invoked after all threads are
  // finished.
  if (runManagerType == sequentialRM) TerminateEventLoop();
}
```

```cpp
// ------------------------------------------------------------------
void G4RunManager::ProcessOneEvent(G4int i_event)
{
  currentEvent = GenerateEvent(i_event);
  eventManager->ProcessOneEvent(currentEvent);
  AnalyzeEvent(currentEvent);
  UpdateScoring();
  if (i_event < n_select_msg) G4UImanager::GetUIpointer()->ApplyCommand(msgText);
}
```

```cpp
bool G4AtlasRunManager::ProcessEvent(G4Event* event)
{

  G4StateManager* stateManager = G4StateManager::GetStateManager();
  stateManager->SetNewState(G4State_GeomClosed);

  currentEvent = event;

  eventManager->ProcessOneEvent(currentEvent);
  if (currentEvent->IsAborted()) {
    ATH_MSG_WARNING( "G4AtlasRunManager::ProcessEvent: Event Aborted at Detector Simulation level" );
    currentEvent = nullptr;
    return true;
  }

  if (m_recordFlux) { m_fluxRecorder->RecordFlux(currentEvent); }

  this->StackPreviousEvent(currentEvent);
  bool abort = currentEvent->IsAborted();
  currentEvent = nullptr;

  return abort;
}
```

To discuss

- GeoModel db storage and versioning
- Digitization: where it needs to be done
- What is expected as output from G4Alg