New Classes for Field Configuration

I. Hrivnacova
IJCLab Orsay (CNRS/IN2P3)

29th Geant4 Collaboration Meeting, Catania,
8 October 2024

Outline

- Motivation
- Key capabilities
- Design & implementation
- Field examples update
- Possible futher integration with the Geant4 field classes

Motivation

- Geant4 is capable of describing and propagating in a variety of fields and a number of field integration methods can be used by users to tune their applications for best performances
  - The users can get inspired by the **FieldSetup** and **FieldMessenger** classes provided in the **extended field examples** category
    - These classes are however integrated within the examples codes
    - Each example demonstartes its use case (magnetic field, electric field, ...) with its own FieldSetup and FieldMessenger classes
      - In total ~ 2K lines of code in the first three field examples
    - Integration in the users applications requires some work
- In the Geant4 VMC framework, used in ALICE and other experimpents, these classes were further developed in order to provide the user a possibility to configure the field with UI commands only
  - However they require a continous mantenance and follow up with the code evolution in Geant4 (adding new steppers, changing the default parameters values)

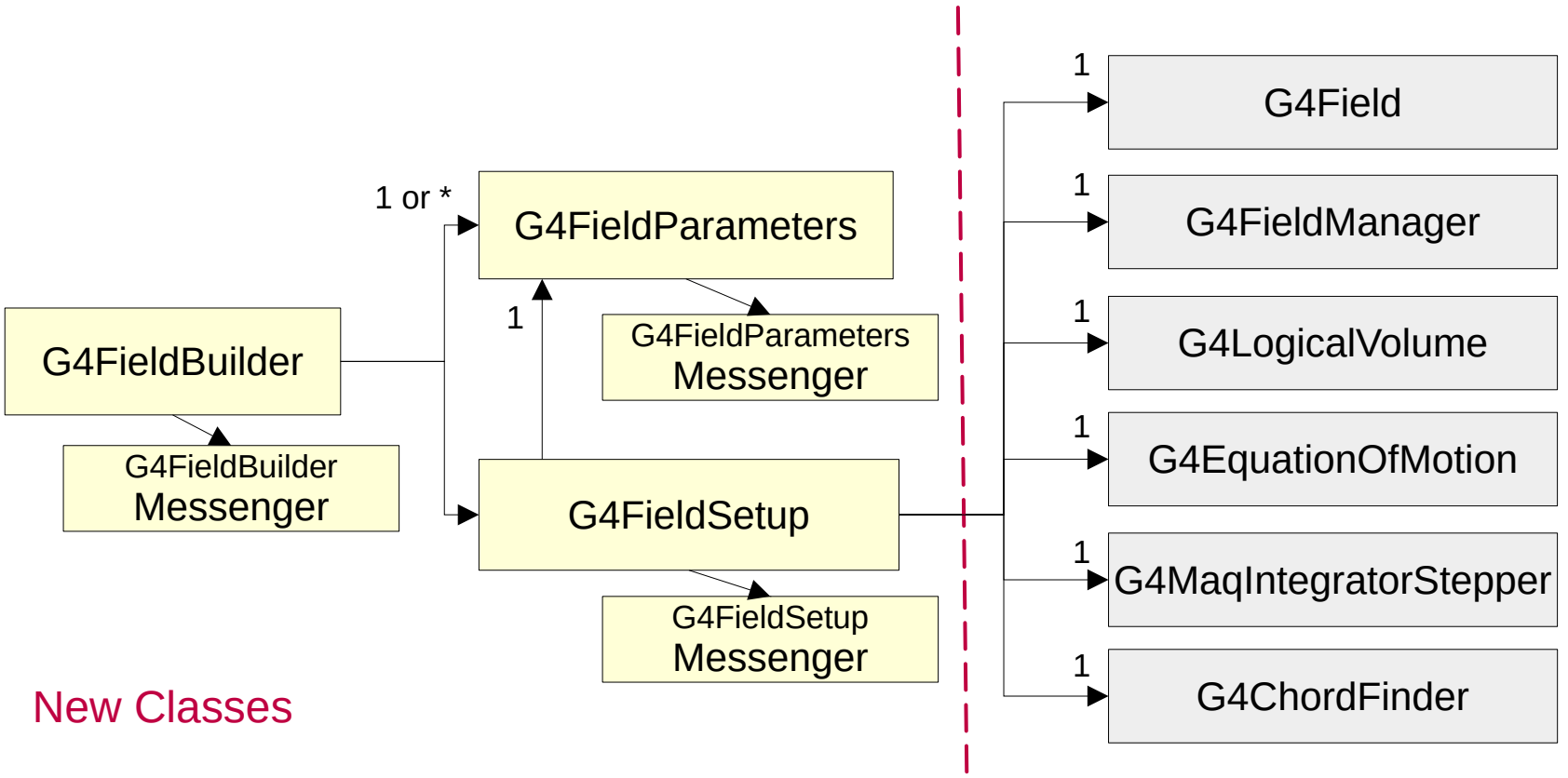
Motivation - 2

- Refactoring the code from the Application framework to Geant4 can be beneficial for all Geant4 users and will move the maintenance to Geant4 field developers
  - The UI commands (now in Geant4) allow the users to change key parameters of field propagation (no user's meesenger)
  - Removed code duplication between multiple examples/applications

Key Capabilities

- A new class for building magnetic or other field using the configuration in field parameters is introduced
- It supports magnetic, electric and electromagnetic fields
- The fields can be global, local or both
- The fields can be cached
- The field parametes, including equation of motion and stepper types can be configured in the user code or via UI commands
  - If a parameter is not set, the default value is used
  - The list of supported features (steppers, equations) can be obtained with the UI commands help
- It covers the most common use cases; but cannot be used to create
  - Equations templated on the field type,
  - Steppers/drivers templated on the equation and field types.

Class Design



New Classes

G4FieldBuilder

- The top class for building magnetic or other field using the configuration in field parameters
- Purpose: Provide a single 'place' to configure field & integration
- The instance of the field builder creates a G4FieldParameters object and the UI commands for customisation of the field default configuration

DetectorConstruction.cc

```
#include "G4FieldBuilder.hh"

DetectorConstruction::DetectorConstruction()
{
  // Create field builder
  // this will create commands for field configuration
  G4FieldBuilder::Instance();
  // G4FieldBuilder::Instance()->SetVerboseLevel(2);
}
```

G4FieldParameters

- The class to define the field configuration:
  - The field type, the type of equation of motion of a particle in a field and the integration method, as well as other accuracy parameters.
  - The default values correspond to the defaults set in Geant4 field classes
- Includes new enumerations G4FieldType, G4EquationType, G4StepperType and static functions for conversion between enum value and a meaningful name
  - These names are then used in the messenger for building UI commands and the list of commands candidates

UI Commands

- After the G4FieldBuilder instatiation, the following commands are available to users:

```
/field/fieldType fieldType
        fieldType = Magnetic | ElectroMagnetic | Gravity
```

```
/field/equationType eqType
        eqType = MagUsualEqRhs | MagSpinEqRhs | EqMagElectric |
                 EMFieldWithSpin | EqEMFieldWithEDM
```

  - The existing names, derived from the Geant4 equation class names, will be replaced with better names:

```
        eqType = EqMagnetic | EqMagneticWithSpin | EqElectroMagnetic  |
                 EqEMfieldWithSpin  | EqEMfieldWithEDM
```

UI Commands -2

- The commands for stepper type selection:

```
/field/stepperType stepperType
         stepperType = CashKarpRKF45 | ClassicalRK4 | ExplicitEuler |
               ImplicitEuler | SimpleHeum | SimpleRunge | ConstRK4 |
               ExactHelixStepper | HelixExplicitEuler | HelixHeum |
               HelixImplicitEuler | HelixMixedStepper | HelixSimpleRunge |
               NystromRK4 | RKG3Stepper
```

UI Commands - 3

- The UI commands to set parameters used in field propagation and general commands

```
/field/setMinimumStep value
/field/setDeltaChord  value
/field/setDeltaOneStep value
/field/setDeltaIntersection value
/field/setMinimumEpsilonStep value
/field/setMaximumEpsilonStep value
/field/setConstDistance value
# general commands
/field/update
/field/printParameters
/field/verboseLevel value
```

Field Construction

- Users create their field as before in DetectorConstruction::ConstructSDandField() and let G4FieldBuilder create all Geant4 objects for field propagation:

```
void DetectorConstruction::ConstructSDandField()
{
  // Create user detector field
  auto magField = new G4UniformMagField(fFieldVector);

  // Set field to the field builder
  auto fieldBuilder = G4FieldBuilder::Instance();
  fieldBuilder->SetGlobalField(magField);

  // Construct all Geant4 field objects
  fieldBuilder->ConstructFieldSetup();
}
```

Local Field(s)

- To create a new set of parameters (configuration) for a local field, users should call the G4FieldBuilder function CreateFieldParameters with the local field root logical volume name as parameter
  - New commands will be located in the `/field/volumeName` directory

DetectorConstruction.cc

```
#include "G4FieldBuilder.hh"

DetectorConstruction::DetectorConstruction()
{
  // Create field builder
  // this will create commands for global field configuration
  G4FieldBuilder::Instance();

  // Create new configuration for a local field in
  // logical volume  "Radiator"
  G4FieldBuilder::Instance()->CreateFieldParameters("Radiator");
}
```

Local Field Construction

- Users create the local field in DetectorConstruction::ConstructSDandField() and let G4FieldBuilder create all field objects

```
void DetectorConstruction::ConstructSDandField()
{
  // Create user detector fields
  auto globalMagField = new G4UniformMagField(fGlobalFieldVector);
  auto localMagField = new G4UniformMagField(fLocalFieldVector);

  // Set fields to the field builder
  auto fieldBuilder = G4FieldBuilder::Instance();
  fieldBuilder->SetGlobalField(globalMagField);
  fieldBuilder->SetLocalField(localMagField, fRadiatorLV);

  // Construct all Geant4 field objects
  fieldBuilder->ConstructFieldSetup();
}
```

Field Examples Update

- The extended examples **field01, field02** and **field03** were updated with usage of new G4FieldBuilder
  - **field01**: demonstration of tracking in magnetic field
  - **field02**: demonstration of tracking in electric field
  - **field03**: demonstration of tracking in global and/or local magnetic fields
- The *FieldSetup and *FieldMessenger classes were removed from the examples code
  - This functionality is now available by new classes in Geant4
- Setting the field value was refactored from the field setup and messenger classes (now removed) into the detector construction and its messenger classes (kept)
- The examples macros were updated with usage of new UI commands

- New classes and the examples update was included in Geant4 11.3.beta: we have time to incorporate *quick* feedback into Release 11.3

Possible Further Integration

- Currently the Geant4 field classes are fully independent from newly added ones
  - The field parameters defaults are hardcoded in the Geant4 code
  - No methods to access the defaults from the user code
- Further changes are envisaged to ensure keeping the builder up to date with the field classes modification
  - Use the G4FieldParameters defaults in the Geant4 field classes so that users get the same behavior whether they create their field configuration from scratch or via the builder
  - Provide the new enum values in the Geant4 field classes implemented in Geant4:

```
virtual G4FieldType G4Field::GetFieldType() const = 0;
virtual G4StepperType G4MagIntegrator::GetStepperType() const = 0;
virtual G4EquationType G4EquationOfMotion::GetEquationType() const = 0;
```

    - This would guarantee that the enumerations would not be forgotten in the builder when a new stepper or an equation is added into the system