Data Model for FCC

Colin Bernet, Benedikt Hegner

FCC SW Meeting
18.09.2014

Driving considerations

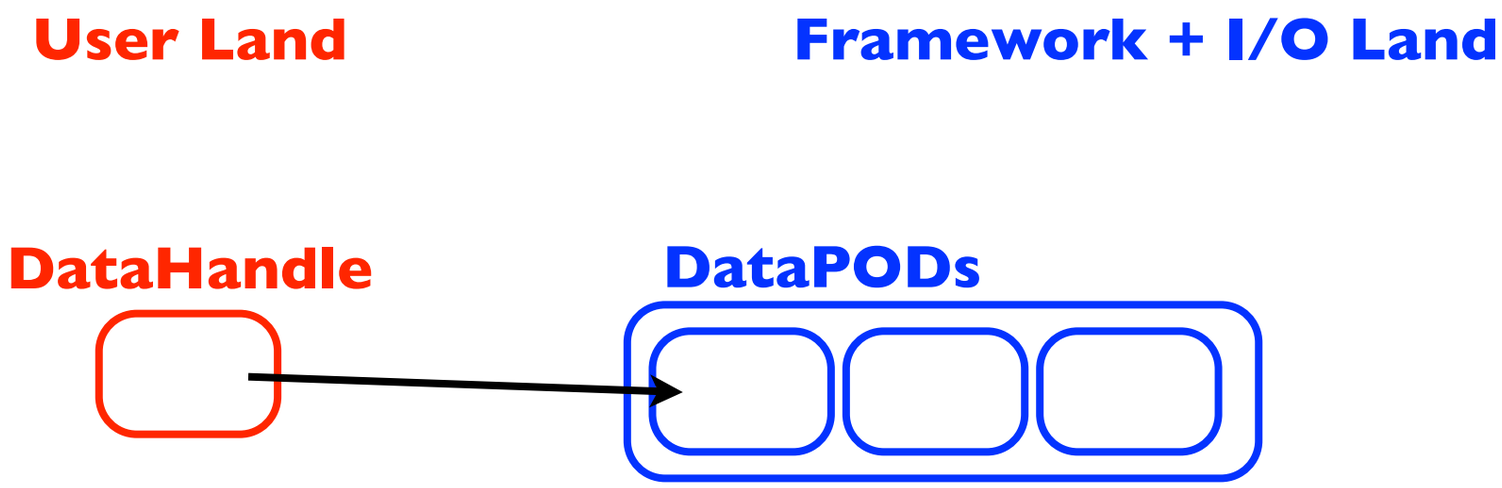
- For FCC studies we want to get the data model right from the ground up
- **ROOT as first choice** for I/O
- No deep object hierarchies
  - Wherever possible **concrete types**
- Simple memory layout
  - Employ **simple structs** instead of fat objects
- We want to have a quick turnaround for improvements
  - **Employ code generation**

Some nomenclature

Preliminary

- **POD**
  - "Plain Old Data"
  - A struct containing the object data
  - Entirely hidden to the user

- **DataHandle**
  - Object given to the user
  - Containing all necessary smartness
  - Redirects all data access to the POD



Code generation

Preliminary

Data definition

```
DummyData:
  Number    : int

ReferencingData:
  DummyData : DummyDataHandle
```

Code generator

Data POD

```
#ifndef DummyData_H

#define DummyData_H

class DummyDataCollection;

class DummyData {
  friend DummyDataCollection;

public:
  const int& Number() const { return m_Number;};
  void setNumber(int& value){ m_Number = value;};

private:

  int m_Number;

};

#endif
```

Example code

Preliminary

```
int main(){
  gSystem->Load("libDataModelExample.so");
  albers::Reader reader;
  albers::EventStore store(nullptr);
  store.setReader(&reader);
  reader.openFile("example.root");
  ReferencingDataCollection* refs(nullptr);
  bool refsPresent = store.get("ReferencingData",refs);

  if (refsPresent){
    std::cout << "Printing ref collection:" << std::endl;
    for(const auto& ref : *refs){
      if (ref.DummyData().isAvailable()) {
        std::cout << "  The Referenced object has the number "
<< ref.DummyData().Number() << std::endl;
      } else {
   std::cout << "  Referenced object not present!" << std::endl;
      }
    }
  }

  return 0;
}
```