An analysis object factory

P. Ronchese
Dipartimento di Fisica e Astronomia “G.Galilei”

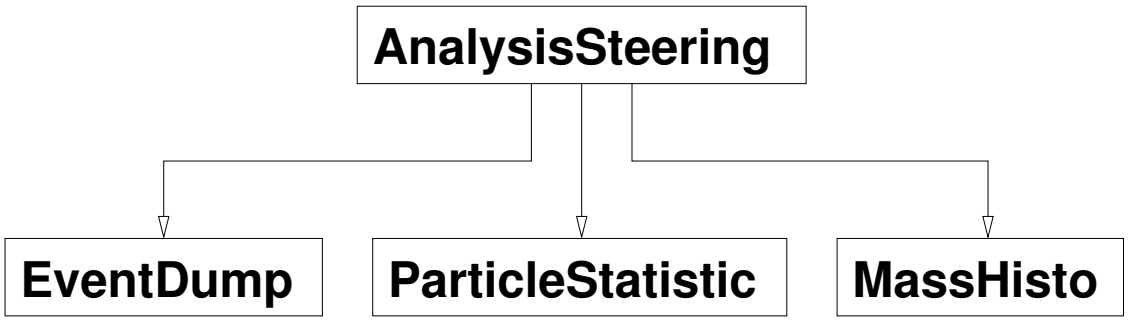
Università di Padova

“Object oriented programming and C++” course

The “analysis factory”

The creation of `EventDump` , `ParticleStatistic` and `MassHisto` objects is delegated to a `class AnalysisFactory` object.

- All analyzers derive from the common `AnalysisSteering` interface.
- The concrete analyzers must be known by the objects creating them.



Requirements

- We want avoid an explicit dependence of `AnalysisFactory` on the concrete analyzers.
- We want avoid the need of modifying `AnalysisFactory` when new analyzers are available.

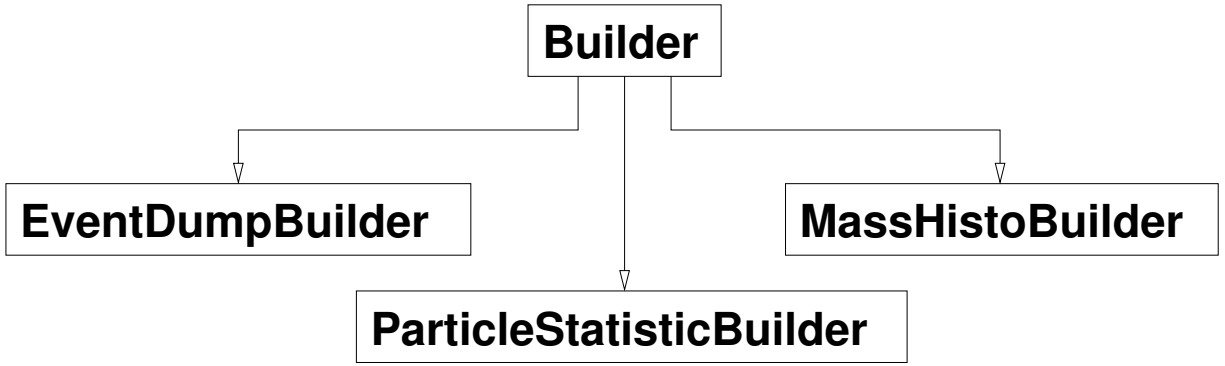
In this way the flexibility of the system is significantly improved.

Analysis builder

We create a `Builder` interface, with a `virtual` function `create` returning a pointer to `AnalysisSteering`.

- It's not a "Builder" in the sense used in the pattern with the same name.
- It's more similar to an "Abstract Factory".
- A concrete builder, e.g. `EventDumpBuilder`:
  - is a `class` deriving from `Builder` ,
  - implements `create` to return a concrete analyzer, e.g. `EventDump` .
- In an analogous way:
  - `MassHistoBuilder` creates a `MassHisto` ,
  - `ParticleStatisticBuilder` creates a `ParticleStatistic` .
- `AnalysisFactory` holds a list of pointers to `Builder`s.

`Builder` list fill



How do we create the `Builder`s and fill the list of their pointers in `AnalysisFactory` ?

- In `AnalysisFactory` we provide a `static` function taking a pointer to `Builder` as parameter, and saving it in the list.
- In the constructor of `Builder` that function is called giving `this` as parameter.
- We create `static` concrete `Builder`s, so that:
  - they're created at the execution start,
  - when they're created, the `Builder` constructor is run,
  - they're automatically registered in the list.

`Builder` names

We want to decide at runtime which analyzers are to be created and run.

- In `AnalysisFactory` we store pointers to `Builder` into a `std::map` , using `std::string`s as key.
- Each `Builder` is identified by its name.
- The function `create` in `AnalysisFactory` selects only `Builder`s whose name is given in the command line, and runs their `create` function.