LHCb Software Object Model

Ideas for discussion

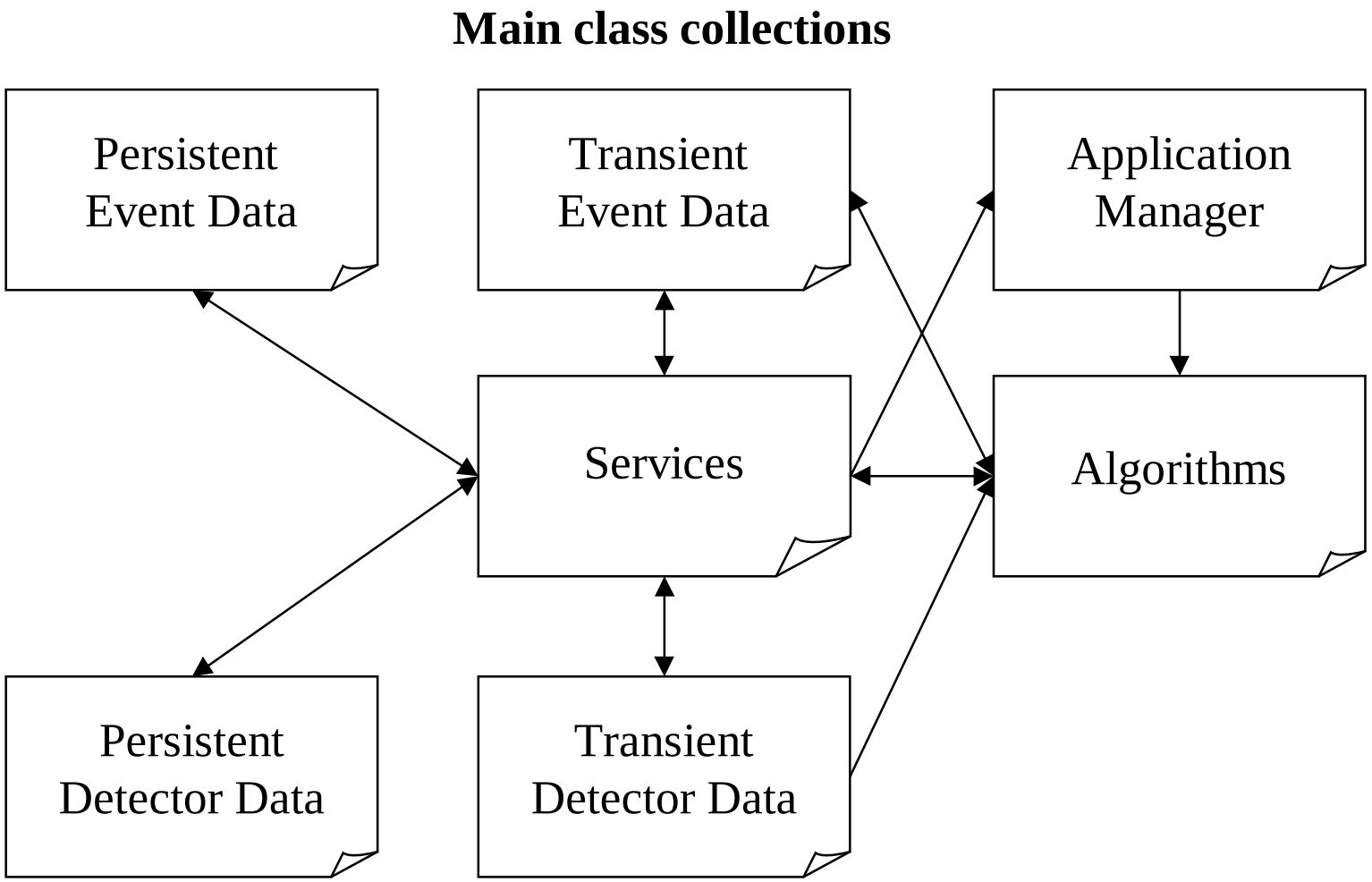
8 September 1998

P. Mato, CERN

Goals and Scope

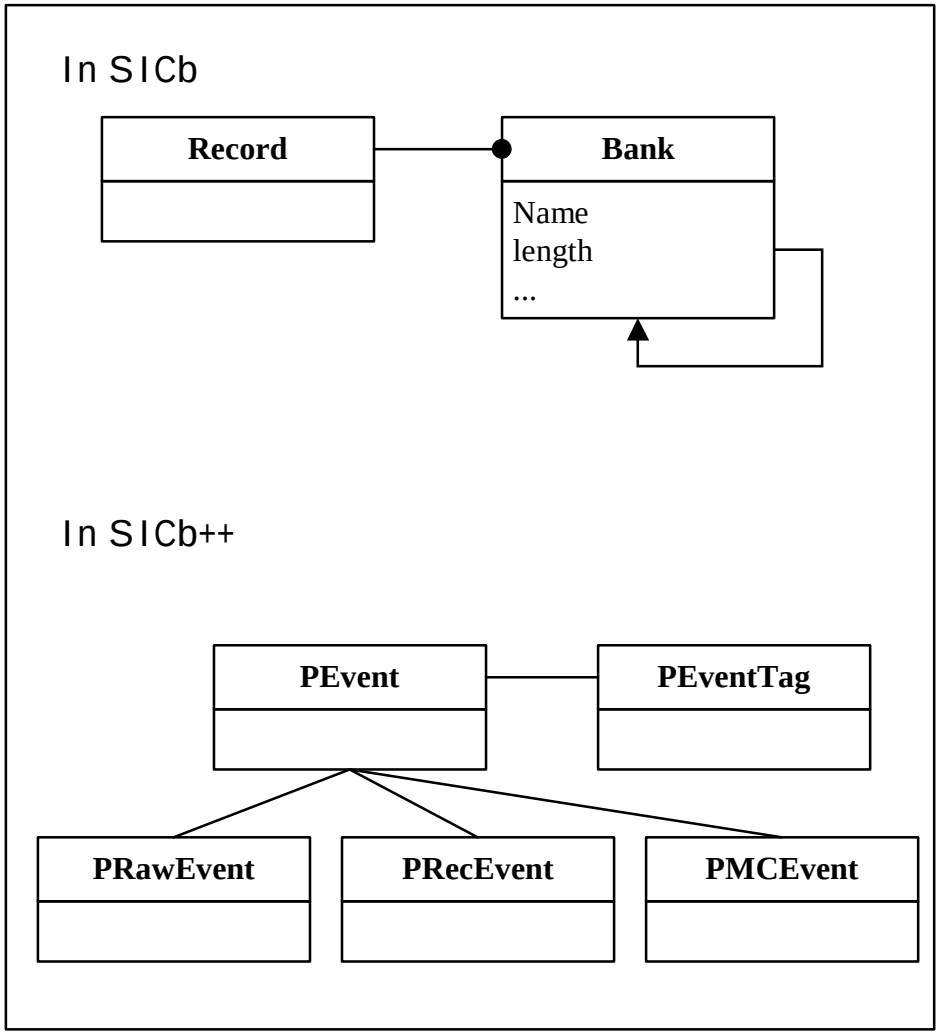
- Goals:
  - Provide a physics analysis framework in C++ to LHCb physicists.
  - Start using the existing data from the current SICb.
- Scope:
  - Start with something simple and perhaps incomplete to understand the problematic.
  - Foresee what locations will accommodate the user code (physicists code) and what kind of interface will be offered.

Domain decomposition



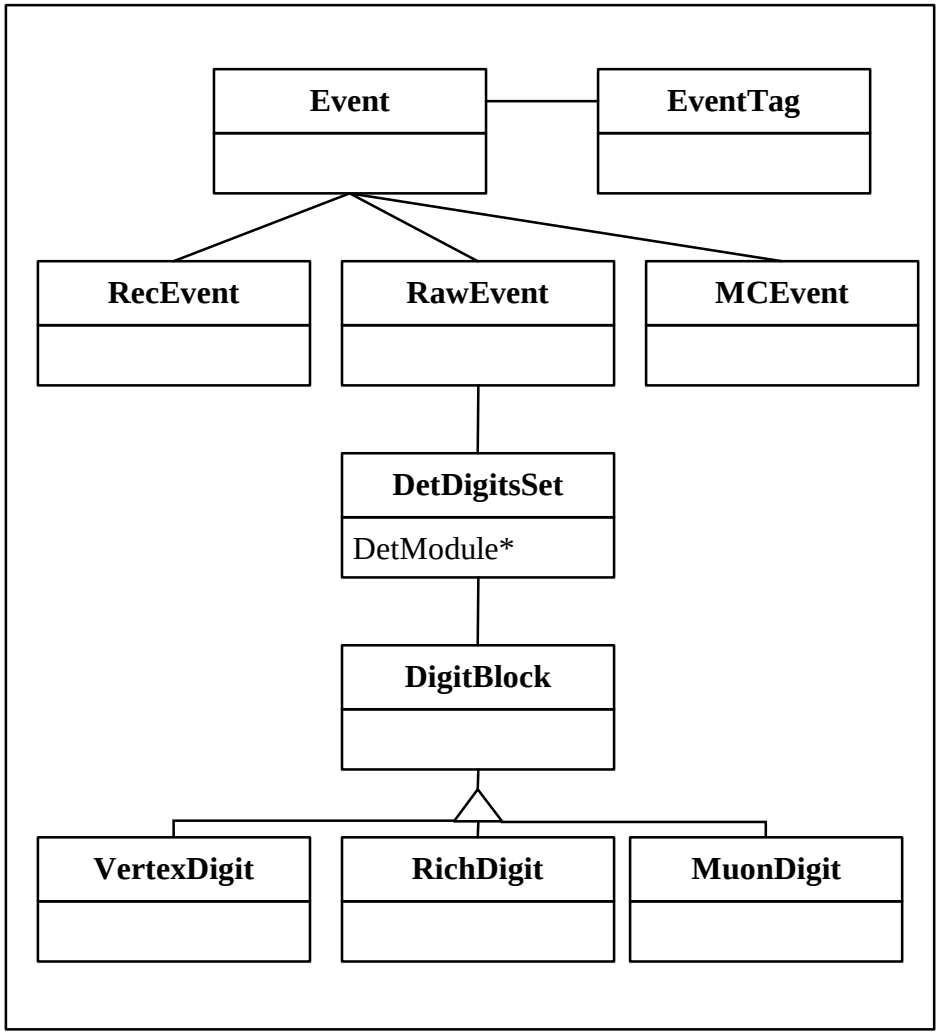
Persistent Event Data

- Are the classes which represent the data model of the data in the **event store**. It includes the raw, monte carlo and reconstructed objects.
- Requirements:
  - Data organized to maximize I/O performance (especially the read performance).
  - Data clustering. Access patterns at the application level.
  - Completeness. Tree like (all entities reachable from single root)
  - Consistent. No duplicated data.



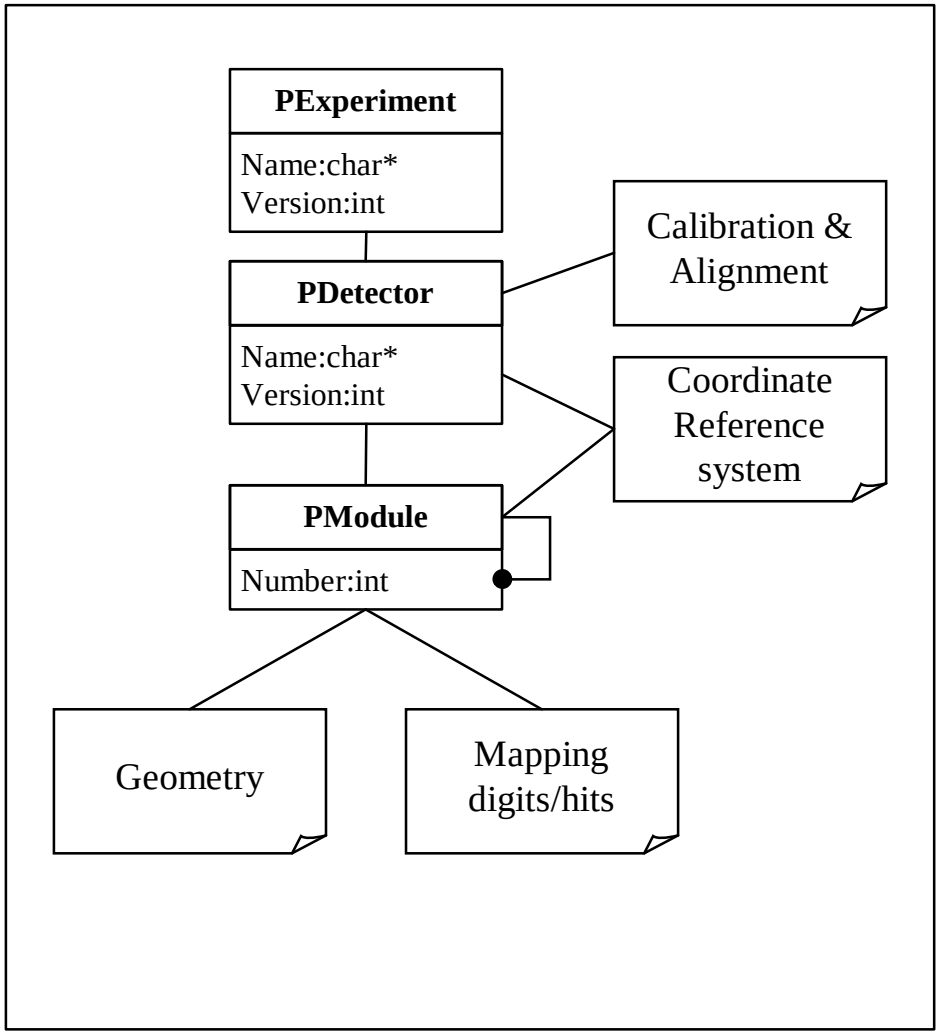
Transient Event Data

- Are the classes which represent the event data in memory to be used by the algorithms. Their lifetime is the time that takes to process one event.
- Requirements:
  - Data is organized to make the life easy for the algorithms.
  - Fast navigation though object relationships. Fast creation and deletion.
  - Data organized to maximize performance during algorithm execution.
  - Data can be duplicated instead of traversing relationships to boost performance.



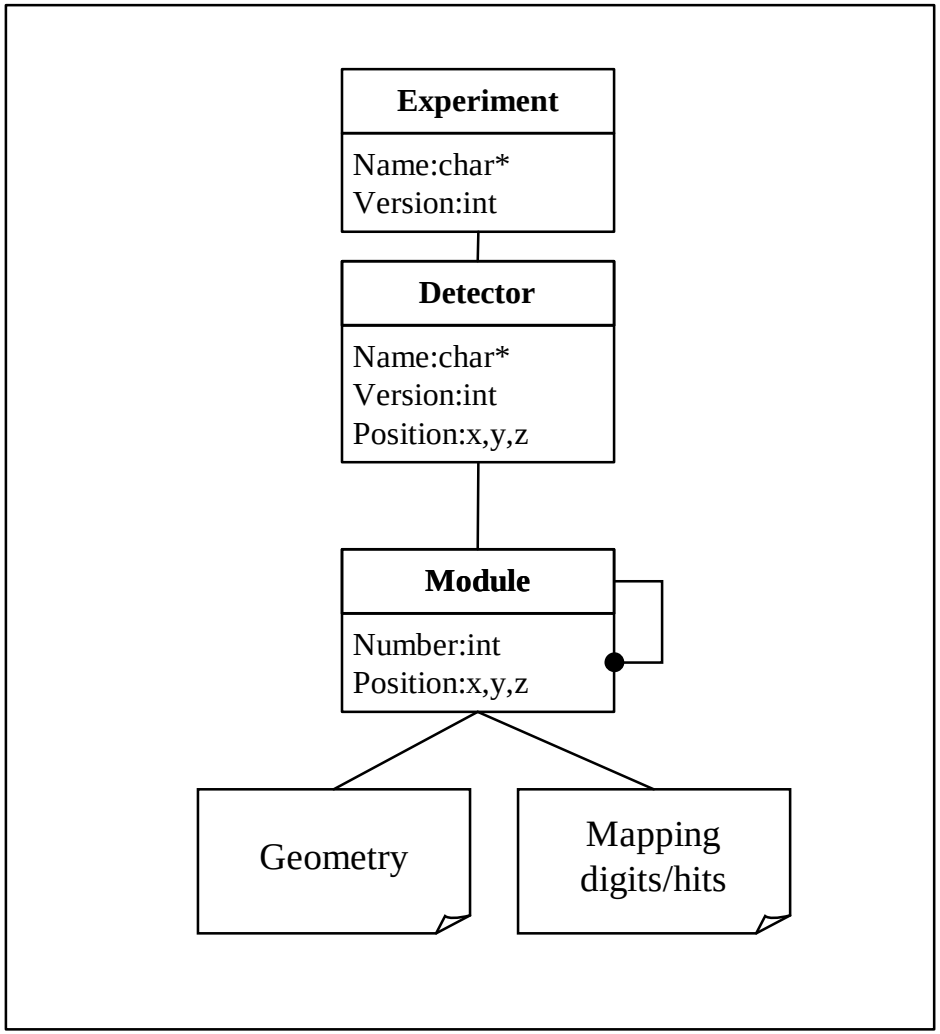
Persistent Detector Data

- Are the classes which represent the data model of the detector data (detector description, geometry, mapping, calibration, slow control, etc.) in the **detector database**.
- Requirements:
  - Coherent and complete set of data. No duplication.
  - Versioning (detector changes, etc.)
  - Validity range (calibration, alignment, etc.). Time or run numbers.



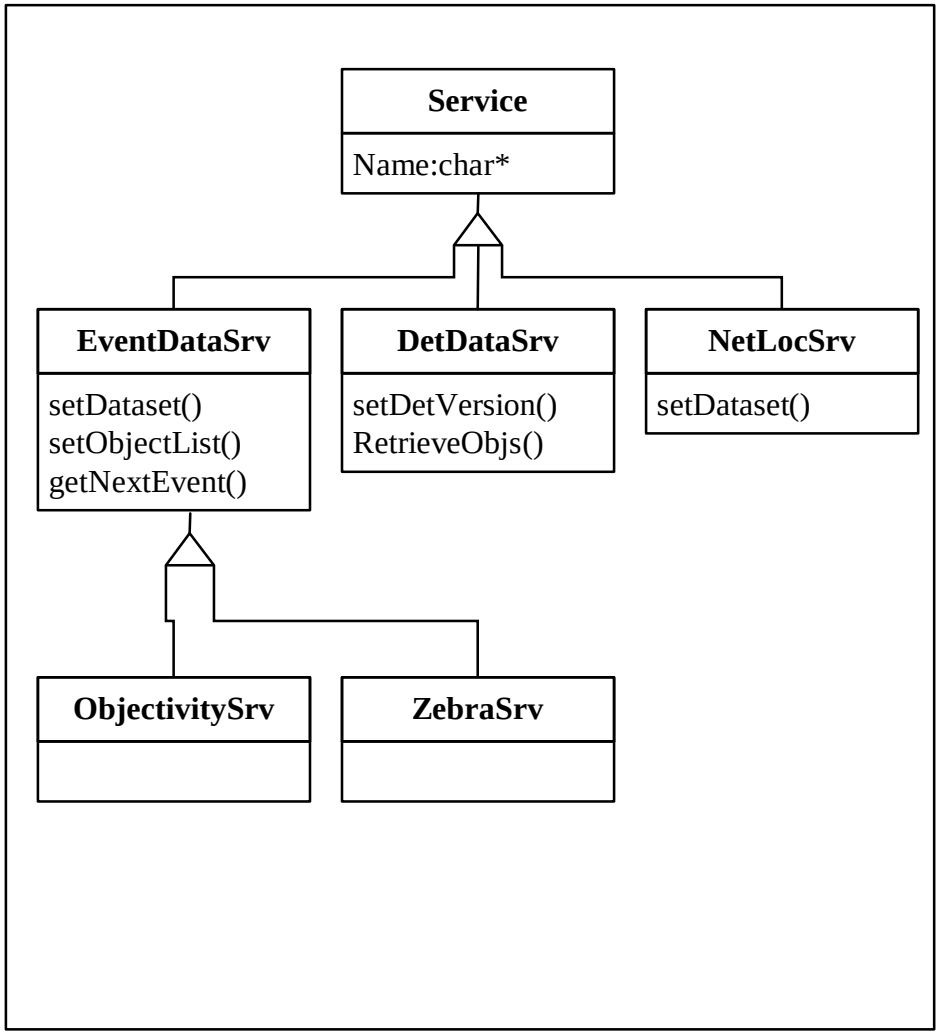
Transient Detector Data

- Are the classes which represent the data model of an **snapshot of detector data** for a given range of events and experiment configuration. The life time is more than one event, usually the complete job.
- Requirements**:**
  - Data is organized to make the life easy for the algorithms.
  - Fast navigation though object relationships.
  - Data can be duplicated instead of traversing relationships to boost performance.



Services

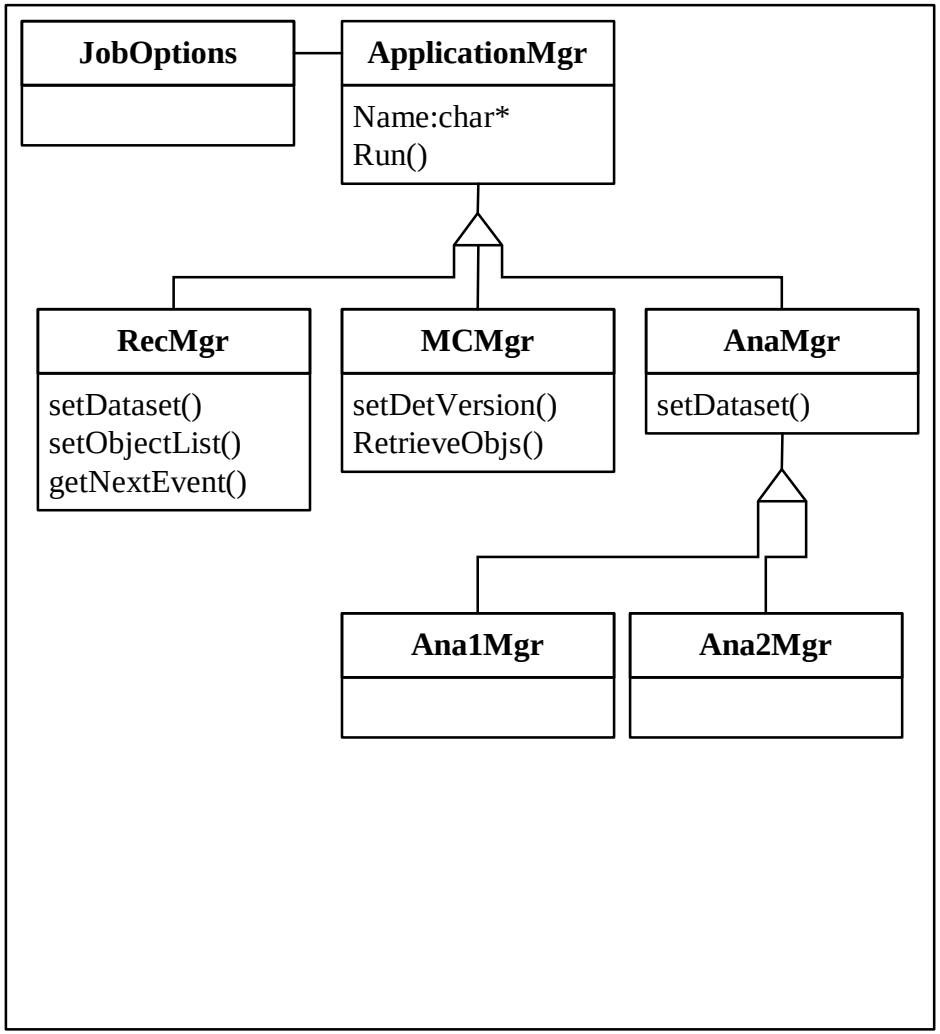
- Are the classes which provides services to algorithms and managers. Examples: “Event Data retrieval service”, “Network locator service”, etc. The life time is usually the complete job.
- Requirements**:**
  - Services are generic and application independent.
  - Algorithms interact to services to access the event and detector data.



Application Manager

- Are the classes which manages a given application type (reconstruction, monte carlo, analysis, etc.) Their life time is the complete job.
- Requirements:
  - One manager per application
  - It sets up the services, algorithms, ... base on the job options and schedules the execution of the algorithms (event loop).
- Main Program:

```
main() {
  RecManager* manager = new RecMgr(…);
  manager.Run();
}
```



Algorithms

- Are the classes which encapsulates chunks of **algorithms**. E.g. “trackfinder”, “kalmanfilter”, “clusterfinder”, etc.
- The algorithms are the place where the “users” can plug their code.
- Their life time is the complete job.
- Requirements**:**
  - Many algorithms per application.
  - Algorithms are selected at run-time based on the job parameters.
  - The application manager orchestrates the calling sequence of the algorithms.
  - Algorithms can be made of other algorithms.