LHCb Software Architecture
GAUDI

11 December 1998
P. Mato, CERN
Project Scope

- We are developing a Framework to be used in ALL the LHCb event data processing applications including all stages: high level trigger, simulation, reconstruction, analysis.

*The Framework are the foundations and the walls of a house that has been designed but still needs to be filled/completed to be fully functional.*
Framework goals

- The bulk of the LHCb data processing software will be developed by physicists.

- The Framework should:
  - Allow physicists to focus on solving the physics problem.
  - Ensure low coupling between concurrent developments.
  - Guarantee a later smooth integration of developments.
  - Facilitate software re-use.
Overall software project planning

- We plan to have major milestones every 2 years. Major project reviews (technologies, methods, quality, etc.)

  - Migration to object-oriented completed.
  - Retirement of the “old” software.

- We plan to go in short cycles (2-3 months) of incremental implementation and release.
  - Feedback from users at each stage.
  - Set priorities for what the following release should contain.
Architecture Design
Major design criteria

- Clear separation between “data” and “algorithms”
- Three basic types of data:
  - **event data** (data obtained from the particle collisions)
  - **detector data** (structure, geometry, calibration, alignment, environmental parameters,..)
  - **statistical data**: (histograms, …)
- Clear separation between “persistent data” and “transient data”.
  - Isolation of user’s code.
  - Different/incompatible optimization criteria.
  - Transient as a bridge between various representations.
Major design criteria (2)

- Data store centered architectural style.
  - Algorithms as data producers and consumers.

- *User code* encapsulated in few specific places:
  - “Algorithms”: Physics code
  - “Converters”: Converting data objects into other representations

- All components with well defined “interfaces” and as “generic” as possible.

- Re-use components where possible

- Integration technology standards
Architecture: Object Diagram
What was not shown
## Architecture: Classification of Classes

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Managers</td>
<td>One per application. The &quot;chef d'orchestra&quot;.</td>
</tr>
<tr>
<td>Services</td>
<td>Offering specific services with well-defined interfaces. Different concrete implementations depending of specific functionality.</td>
</tr>
<tr>
<td>Algorithms</td>
<td>Physics code. Nested algorithms. Simple and well defined interface.</td>
</tr>
<tr>
<td>Converters</td>
<td>In charge of converting specific event or detector data into other representations.</td>
</tr>
<tr>
<td>Selectors</td>
<td>Components to process a selection criteria for events, parts of events or detector data.</td>
</tr>
<tr>
<td>Event/Detector data</td>
<td>The data types that the algorithms and converters are using. No complex behaviour.</td>
</tr>
<tr>
<td>Utility classes</td>
<td>All sort of utility classes (math &amp; others) to help on the implementation of the algorithms.</td>
</tr>
</tbody>
</table>
Architecture (class diagrams)

Services

- PersistencySvc
- JobOptionSvc
- ObtyPersSvc
- ZebraPersSvc
- MessageSvc
- EvtDataSvc
- IConversionSvc
- IPersistenceSvc
- IQueryOptions
- IMessage
- IDManager
- IDProvider
- IService
Architecture (class diagrams)

Algorithms

- Algorithm
  - TrackFinder
    - KalmanFilter
    - AnotherTrkFinder
  - ClusterFinder
  - GenericAlgorithm
    - SelectionAlg
- IProperty
- IAlgorithm
- IGenAlgorithm
- ISelect
Transient Event Store

Persistency Service

Fetch() Store()

creates

Event Data Service

Find(“EcalDigits”, 4)
Register(“key”, #)

Algorithm

Direct reference

Transient Event Store

Event

AnaEvent

MCEvent

RawEvent

Event

Ecal

EcalDigits(1)

EcalDigits(2)

EcalDigits(3)

EcalDigits(4)

HCal

ITrack

Muon

OTrack

Rich

Vdet

RecEvent

Event

LHCb Computing
Each Algorithm only knows what data (type and name) is expecting as input and creating as output.

The only coupling is through the data.

Scheduling of sub-algorithms is responsibility of the parent algorithm.
Algorithm code example

```
StatusCode TrackFitter::execute() {
    // Get the vertex hits from the EventDataSvc
    StatusCode sc;
    DataObject *pDO = 0;
    sc = eventDataSvc->findObject("/Event/RawEvent/Vdet/Station[1]/Clusters", pDO);
    // Check that the data was found
    if(SUCCESS != sc) {
        messageSvc->logFatalError("Cluster data not found in store");
        return sc;
    }
    // Data was found, cast to correct type
    ObjectSet<Cluster*>* clusters = dynamic_cast<ObjectSet<Cluster>* >(pDO);
    // Create a track container object
    ObjectSet<Track*>* *tracks = 0;
    tracks = new ObjectSet<Track*>("Tracks");
    // Use the clusters to produce Tracks
    // and place the tracks in a container
    tracks->push_back( .. );
    // Register the Tracks with the EventDataSvc
    sc = eventDataSvc->registerObject("/Event/RecEvent/Vdet", tracks);
    // If the registration fails some cleanup must be done
    if(SUCCESS != sc) {
        messageSvc->logFatalError("Failed to register tracks");
        delete tracks;
        return sc;
    }
    return SUCCESS;
}
```
Transient/Persistent Data representations

- Storage MS
- Persistency Service
- Converter
- Selector
- Transient Event/Detector Store
- Data Service
Detector Data representations

- Detector Elements need to be "identified"
- Strong hierarchical structure
- Versioning
Visualization

- Transient Event/Detector Store
- Rep. Service
- Converter
- Selector
- User Interface
- Selects objects in store
- Representations Store (graphical, textual)
Detector Description

- It includes:
  - Detector structure (final detector, test beam, etc.)
  - Geometry & Positions (Ideal, Real, Simulation). Versioning based on time, run #, etc. Material.
  - Mapping electronic channels to detector cells. Dead channels.
  - Detector control data needed for reconstruction (time based).
  - Calibration and alignment data.

- The transient detector store contains a “snapshot” of the detector data valid for the event currently being processed and a labeled version.
Detector Database

Geometry

- Version C
- Version B
- Version A

Alignment

- Version C
- Version B
- Version A

Calibration

- Version B
- Version A

version B at time T
Links between Event/Detector

- A priori different “persistent stores”. Logical identification needed.
Application Configuration

◆ What are the knobs at our disposal?
  – **JobOptions.** Simple usage. It allows the end-user to overwrite any property of any algorithm or service.
  – **Algorithm/Service properties database.** A more sophisticated way to modify the properties of the algorithms and services.
  – **Detector database edition** to create new versions or releases.
  – **Write specific code.** Configure your application by setting it at runtime.
  – **User interface component.** Graphical (a la Visual Basic), command line (scripting language), etc.