
Architecture Review

26 November 1998

P. Mato, CERN

Project Scope, Context and Aims

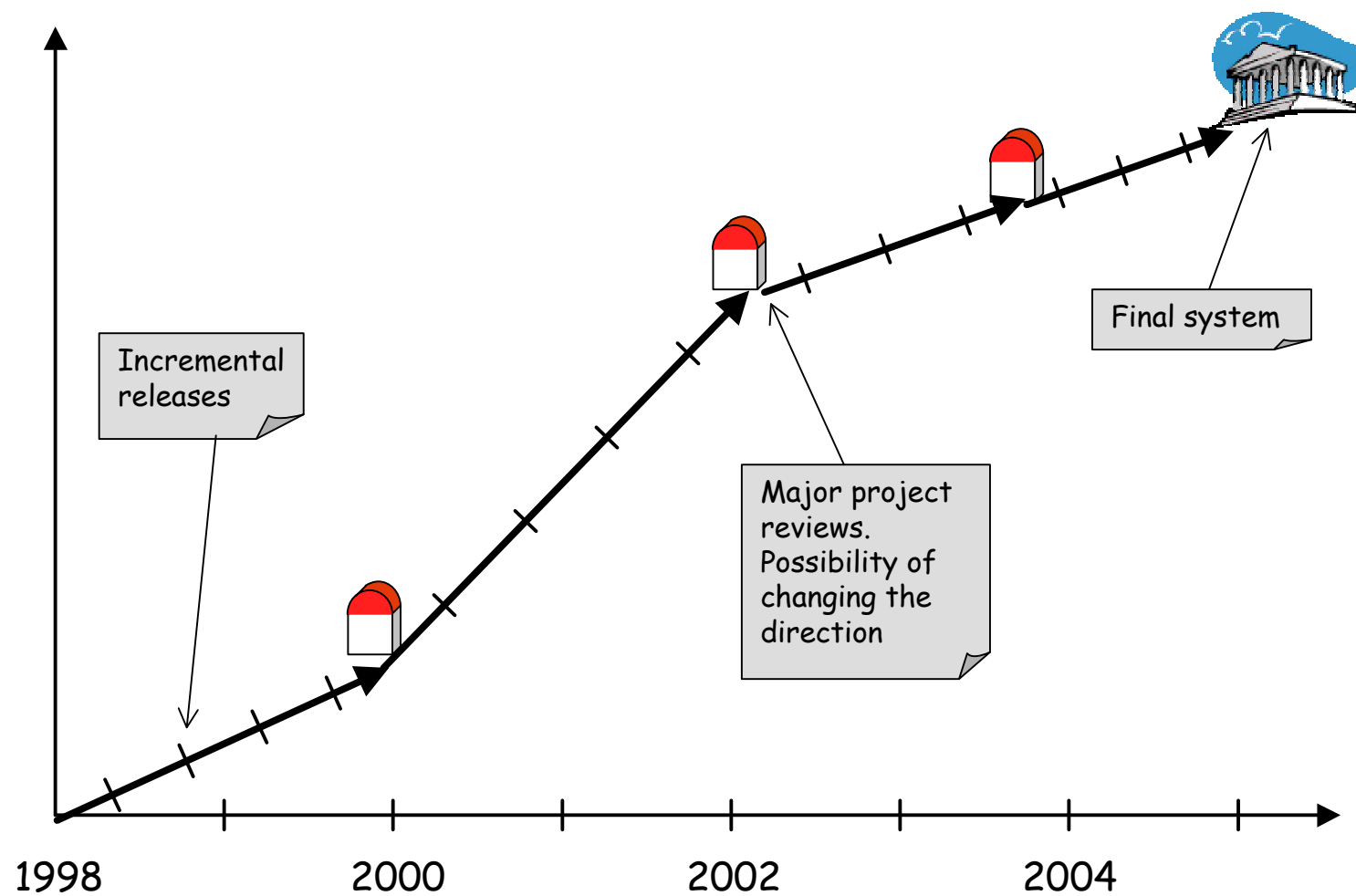
Project Scope

- ◆ We want to develop a Framework to be used in ALL the LHCb event data processing applications including all stages: high level trigger, simulation, reconstruction, analysis.
- ◆ The bulk of the LHCb data processing software will be developed by physicists. The Framework should:
 - Allow them to focus on solving the physics problem.
 - Ensure low coupling between concurrent 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.)
- ◆ First big milestone: mid 2000.
 - 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.

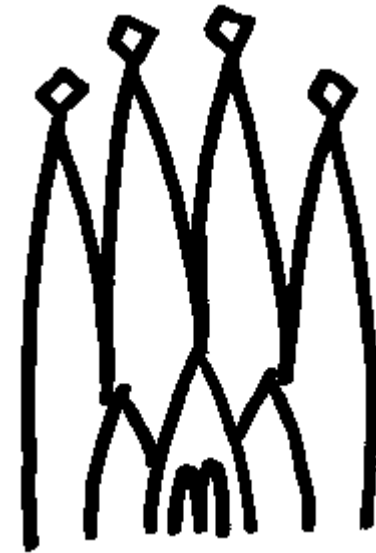
Road map



Release 1.0

- ◆ Functionality release 1.0:
 - Object Oriented environment that allows a user to:
 - » Define input and output data, job parameters (c.f. *SICB.dat*)
 - » Loop over events
 - » For each event, access MonteCarlo truth and digitised raw data
 - » Output results in the form of HBOOK histograms and/or ntuples
 - » Provide placeholders user initialisation and analysis code (c.f. *suinit*, *suanal*)
 - Does NOT allow user to:
 - » Store back into ZEBRA store (*can be discussed...*)
 - » Access SICB reconstruction output
 - » Use an analysis library (c.f. *AXLIB*)
 - Input is from ZEBRA files produced by SICB

System Design Issues



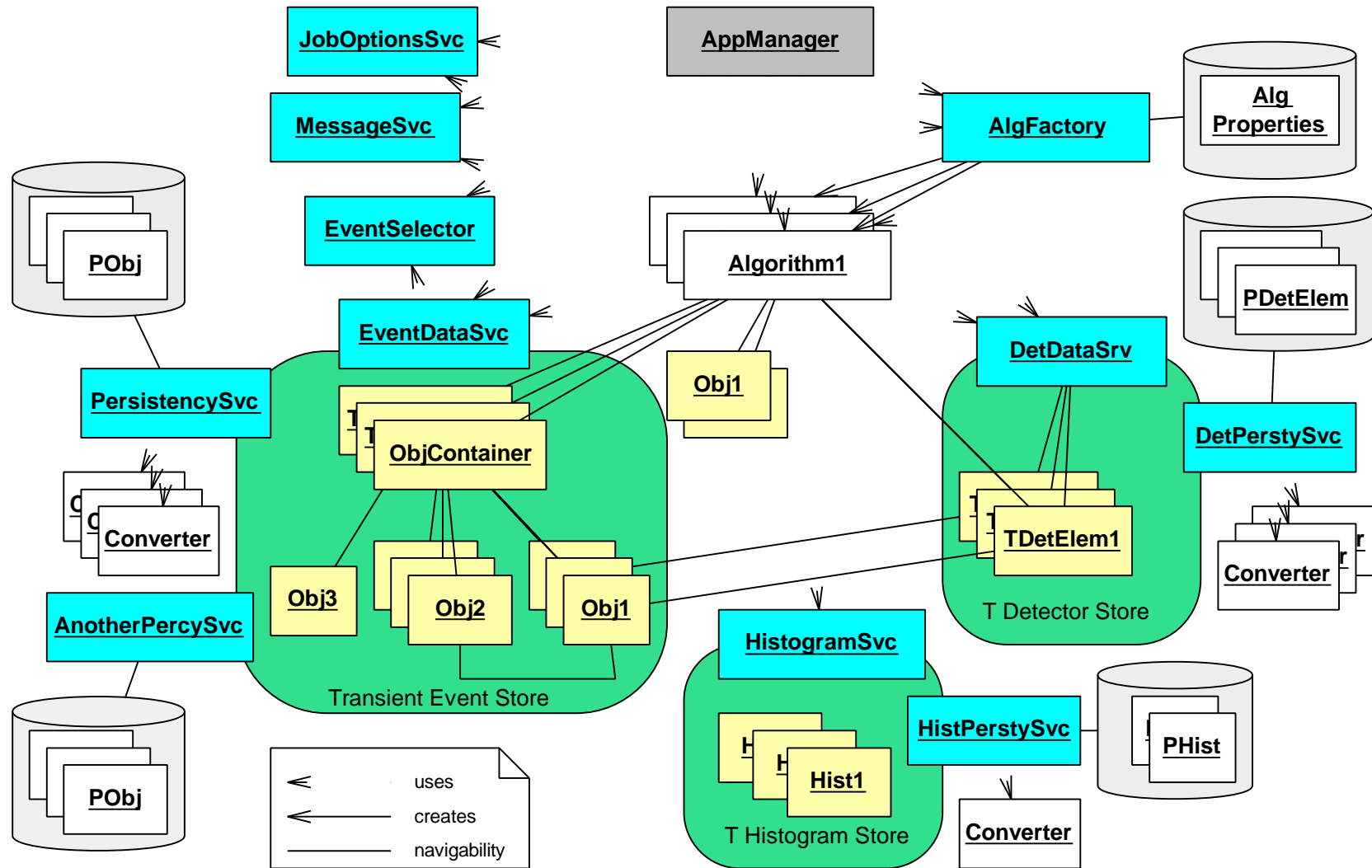
Major design criteria

- ◆ 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.

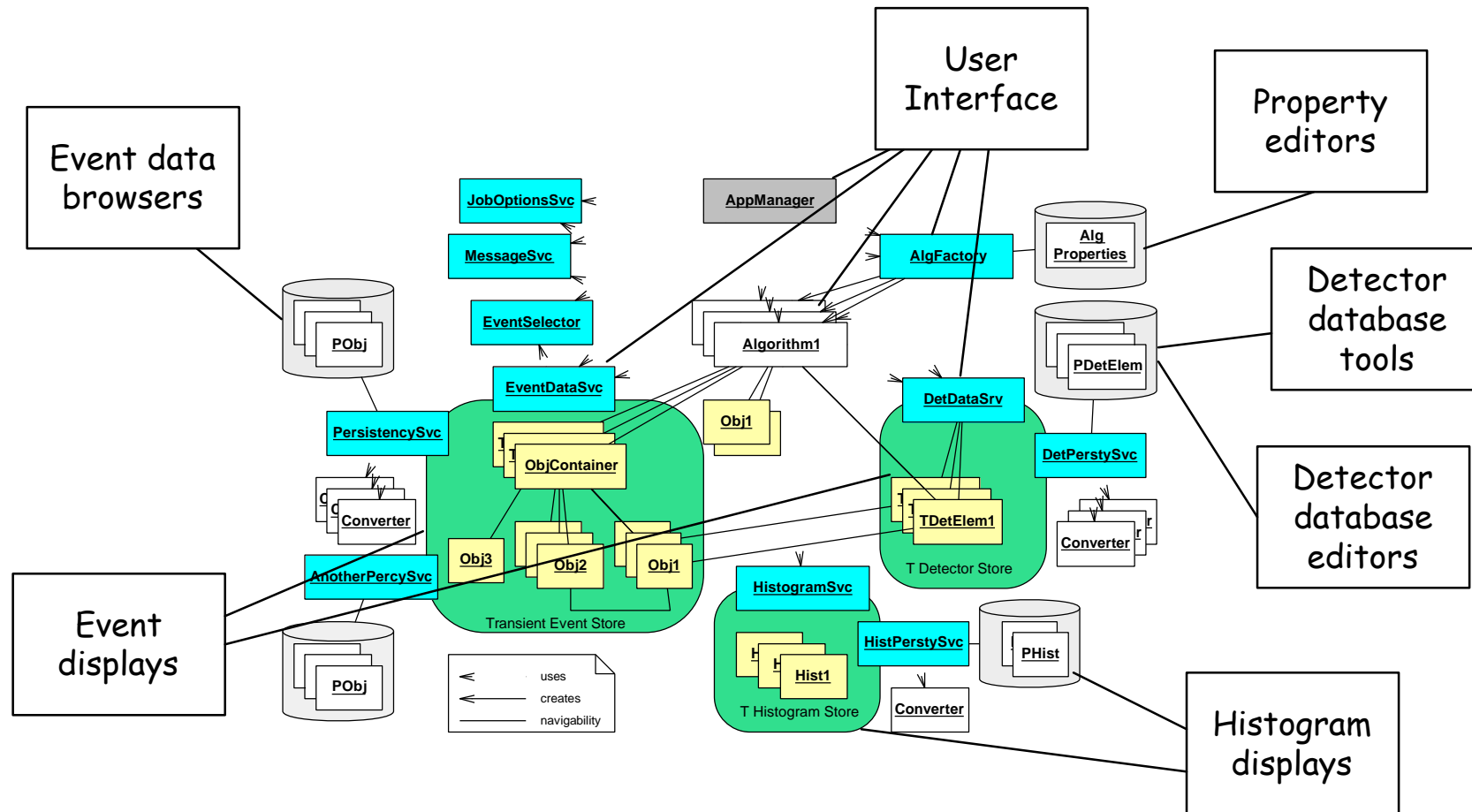
Major design criteria (2)

- ◆ Data 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

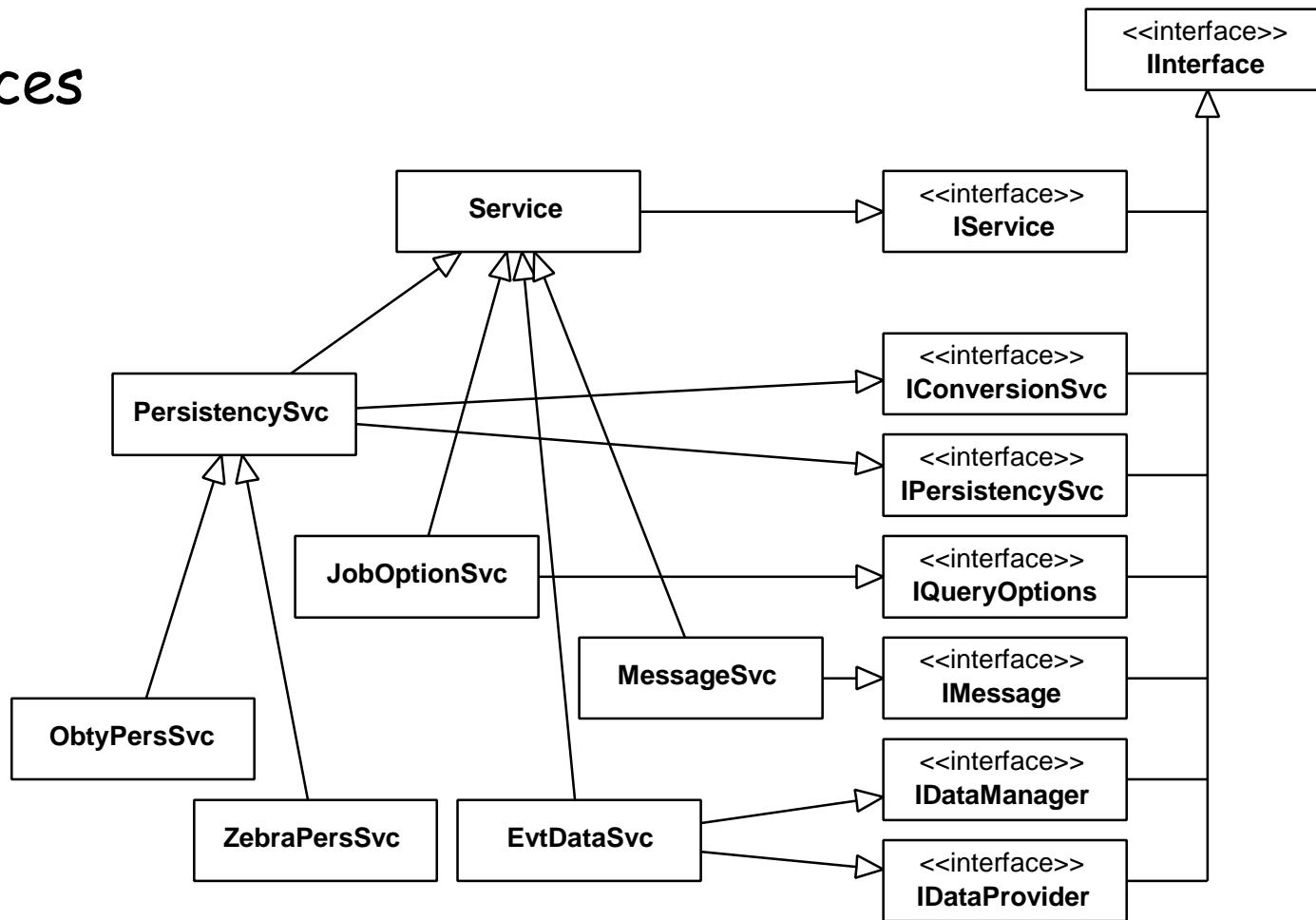


Architecture: Classification of Classes

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

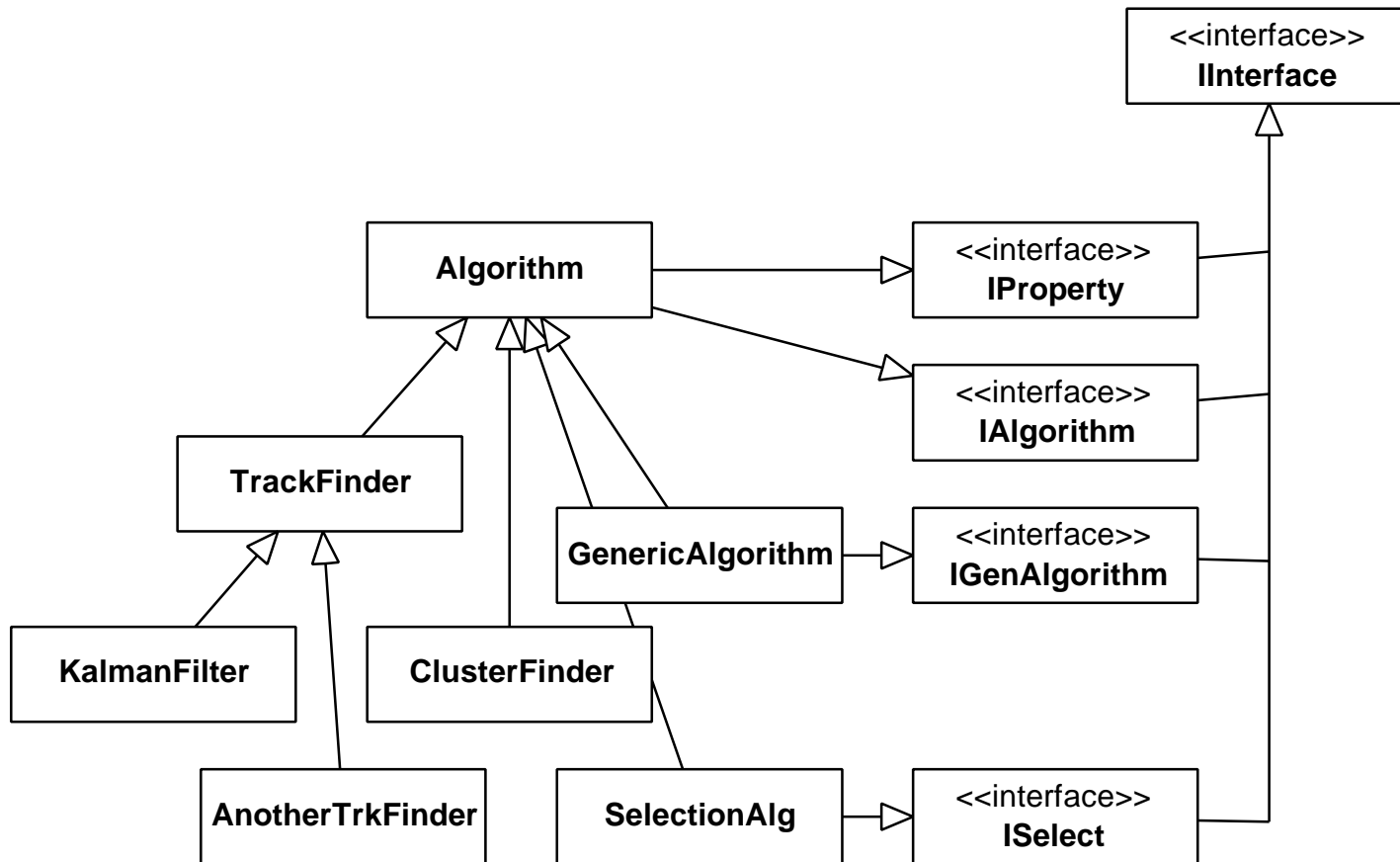
Architecture (class diagrams)

Services

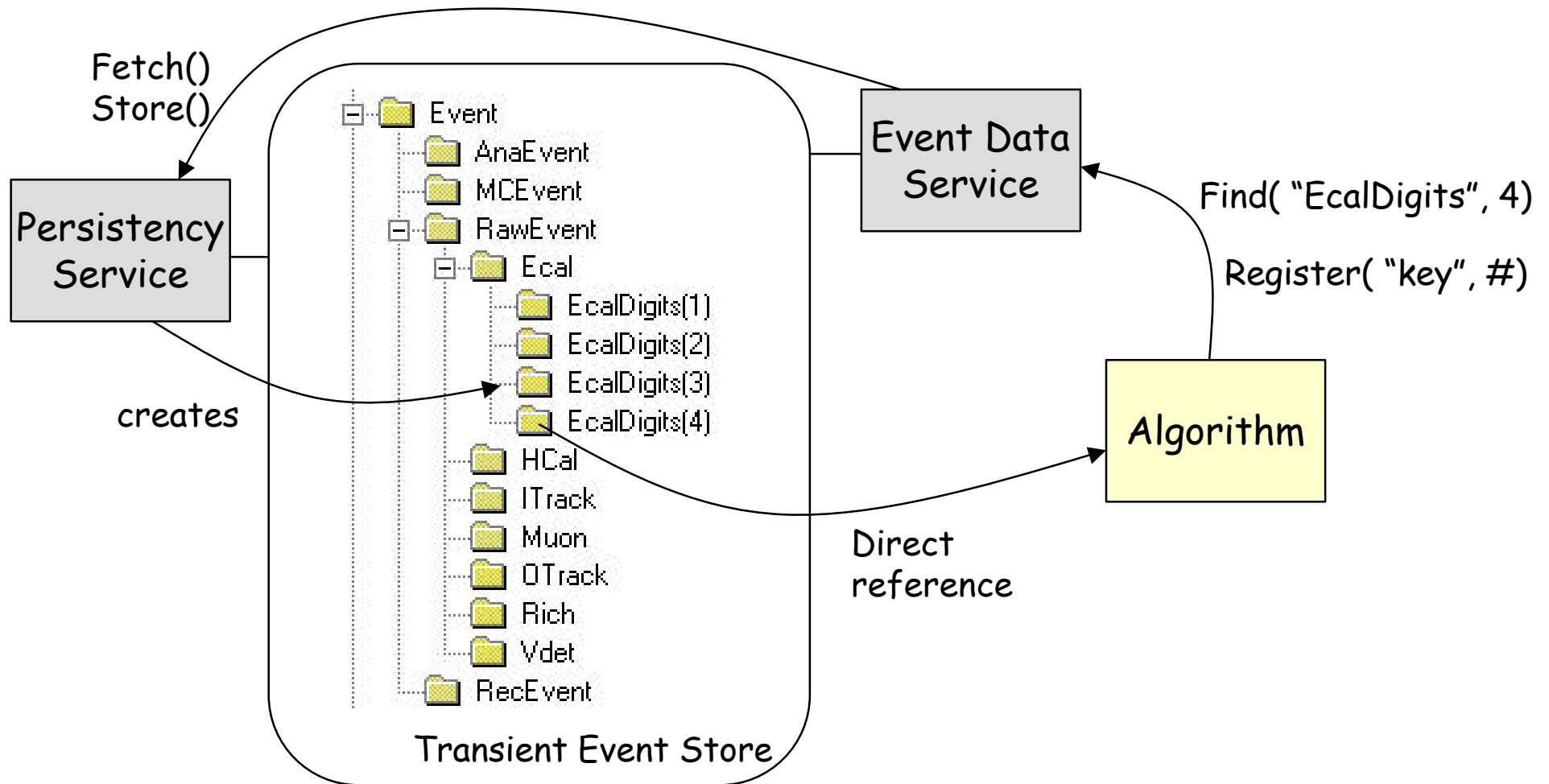


Architecture (class diagrams)

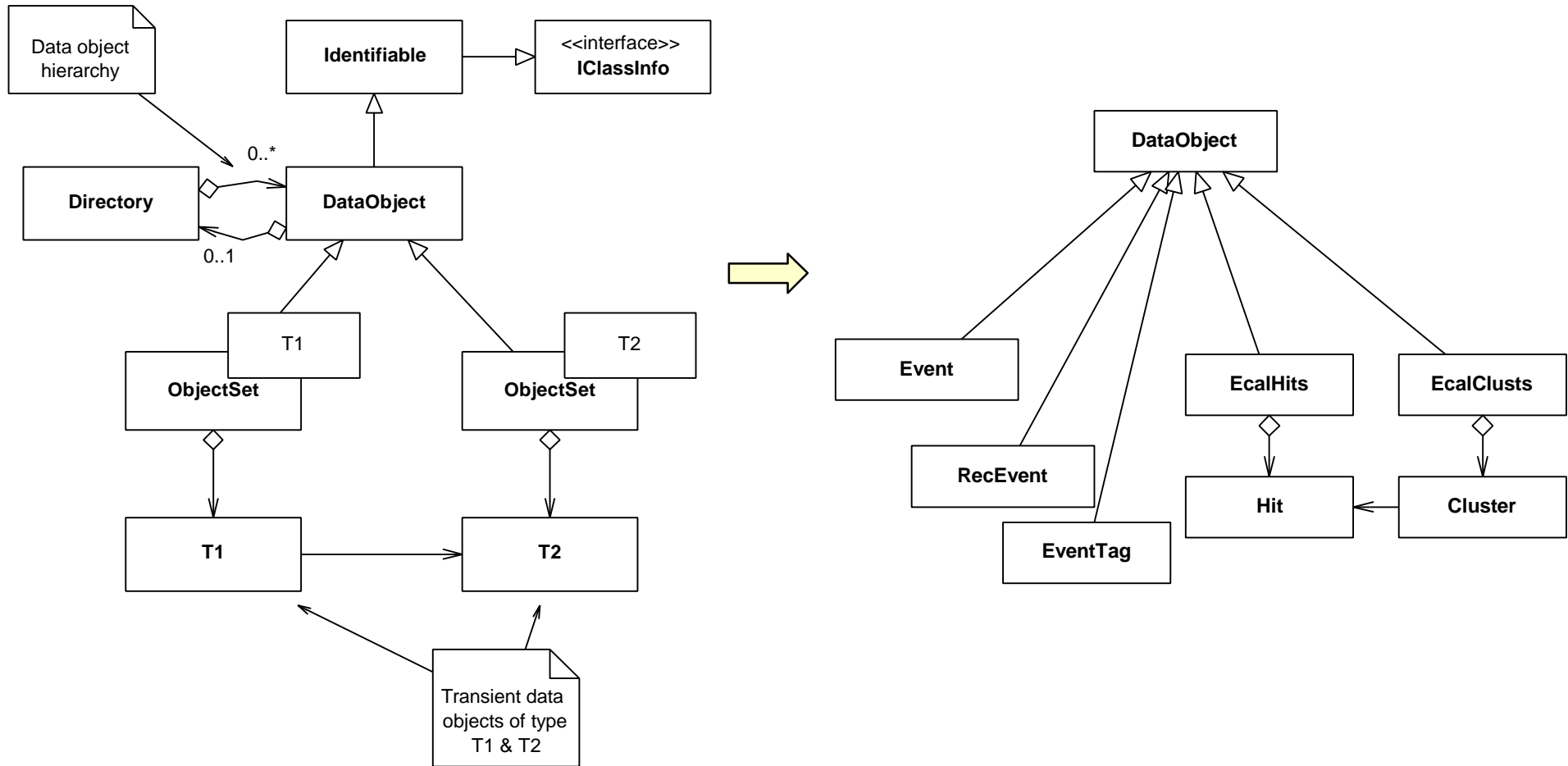
Algorithms



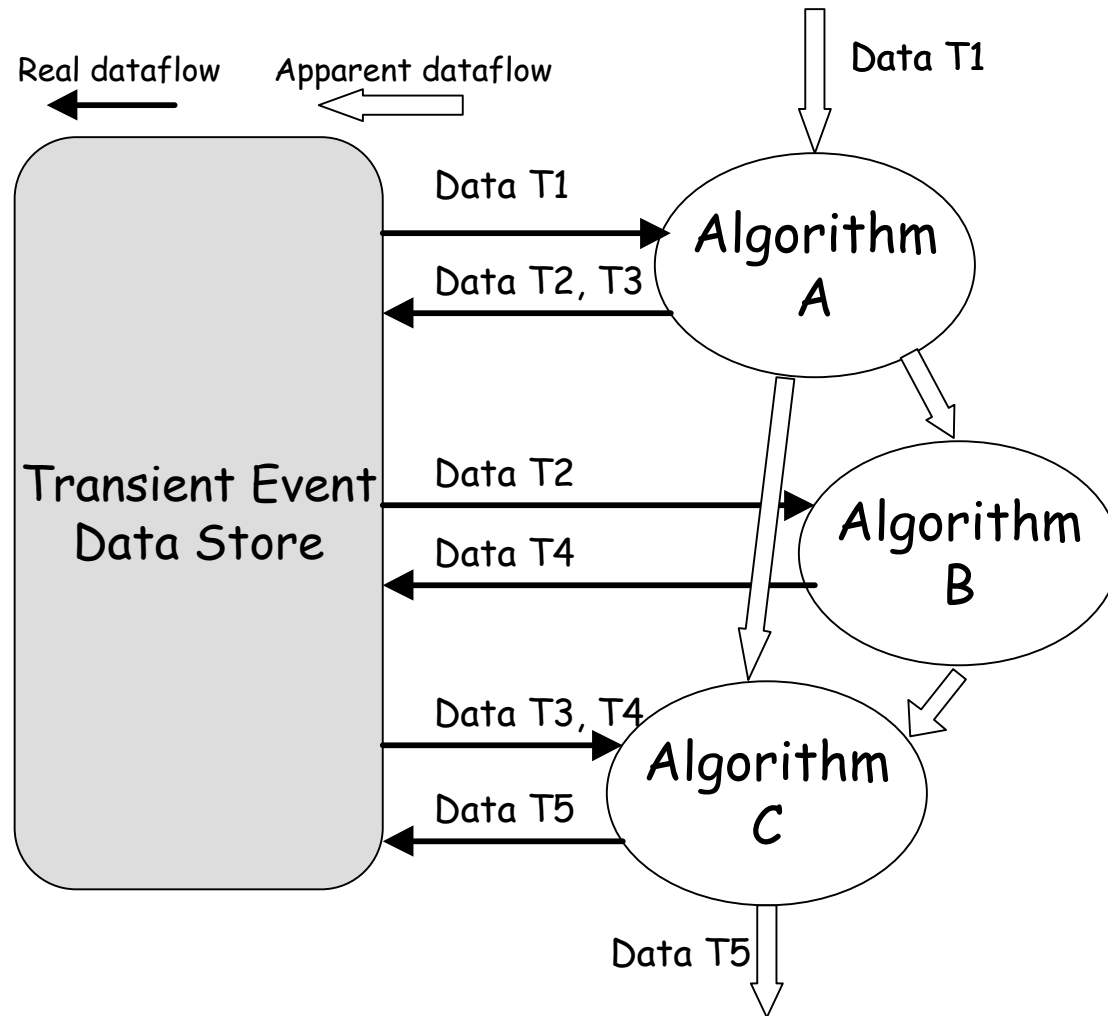
Transient Event Store



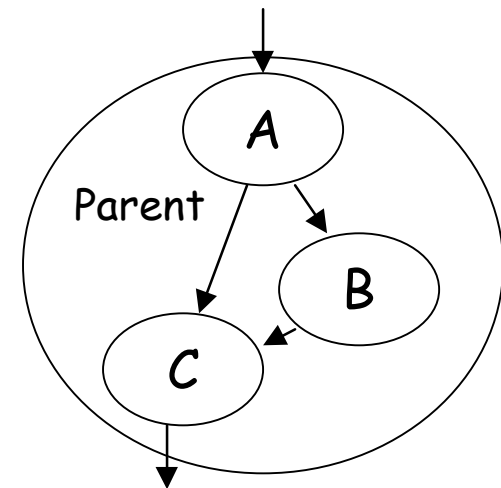
Transient Data Model



Algorithms & Transient Data Store



- 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.

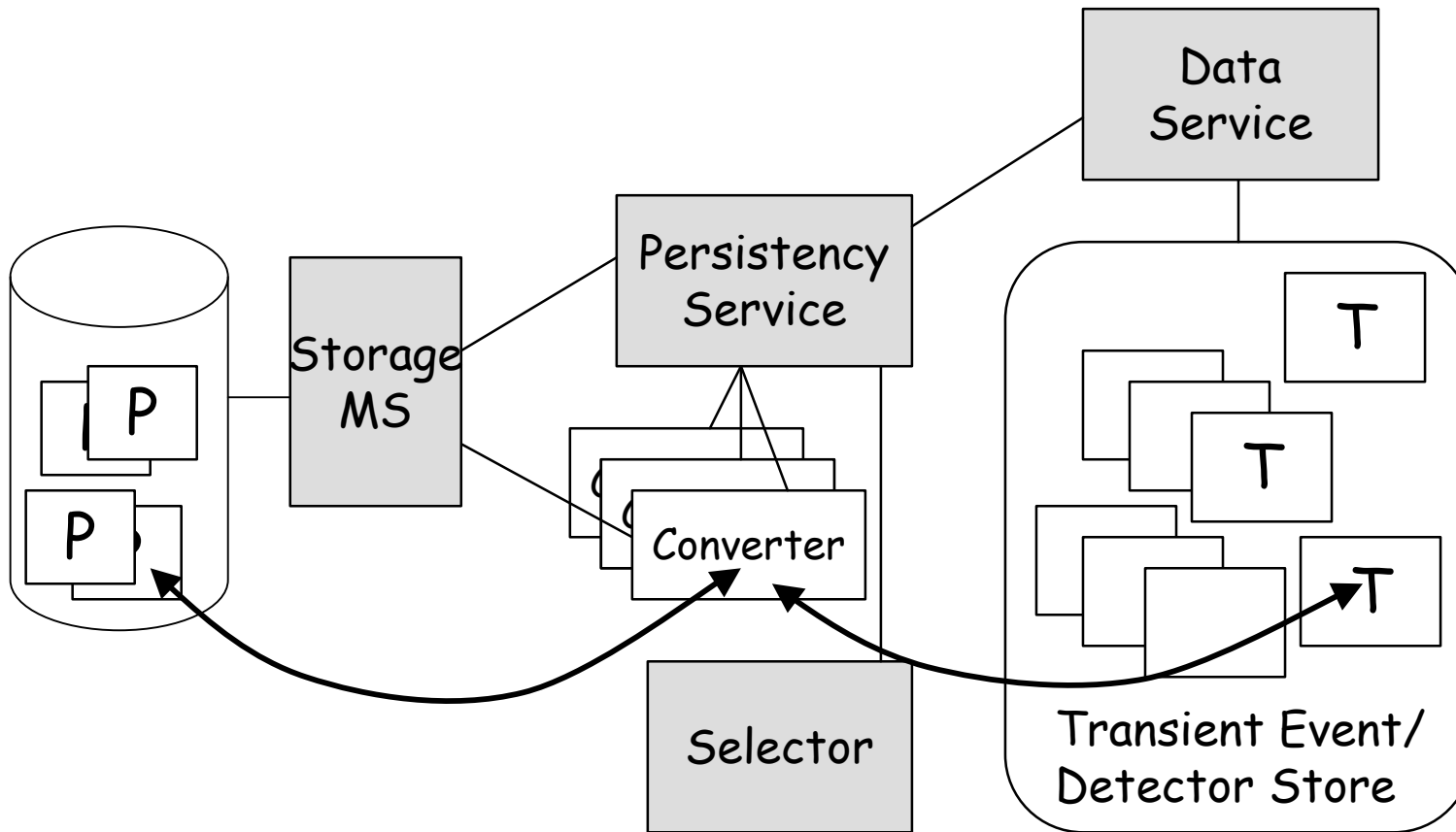


Algorithm code example

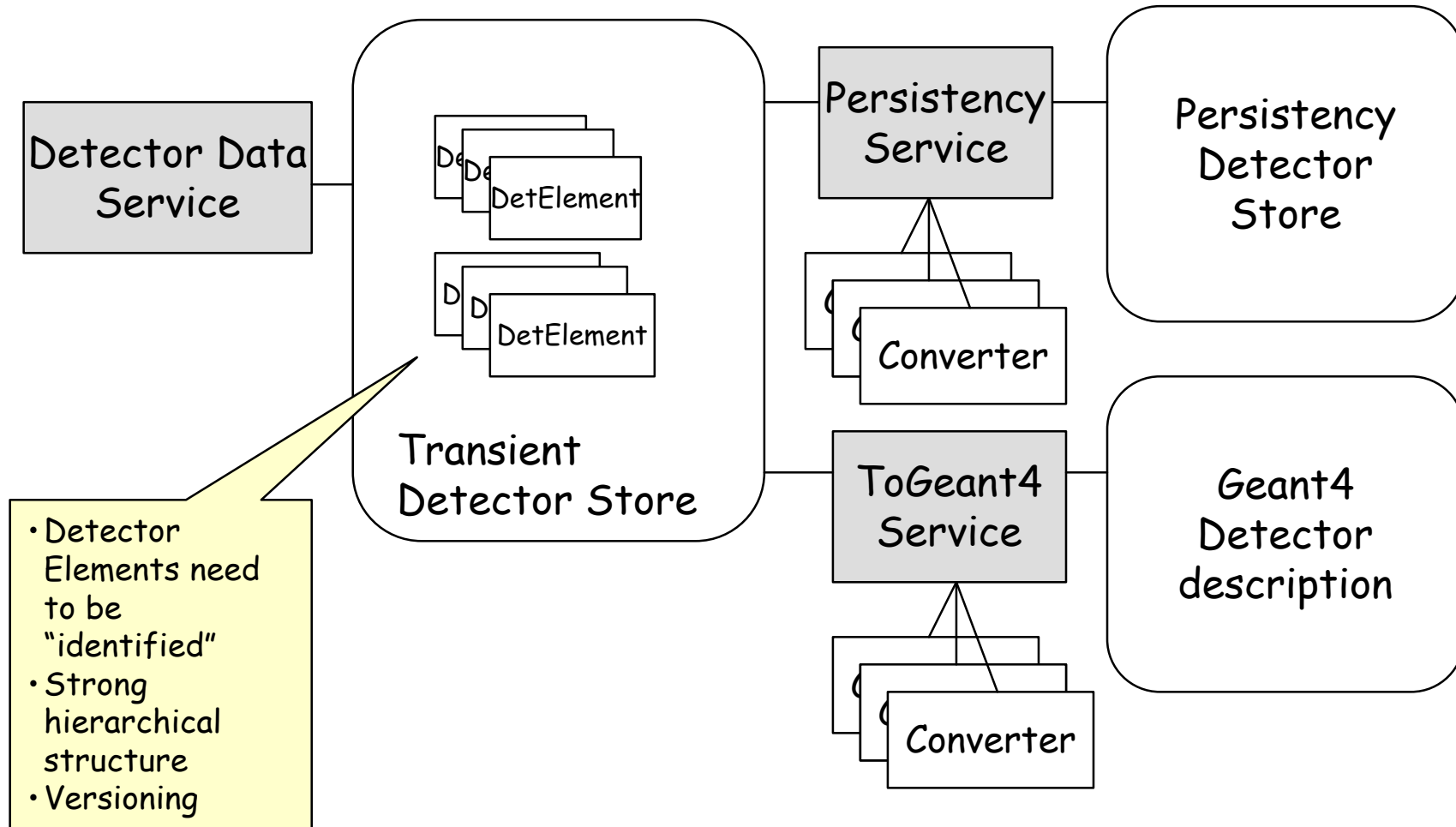
```
StatusCode TrackFitter::execute() {
    // Get the vertex hits from the EventDataSvc
    StatusCode sc;
    DataObject *pDO = 0;
    sc = eventDataSvc->findObject("Event/RawEvent/Vdet/Station[1]/Clusters", pDO);
    // Check that the data was found
    if(SUCCESS != sc) {
        messageSvc->logFatalError("Cluster 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;
}
```

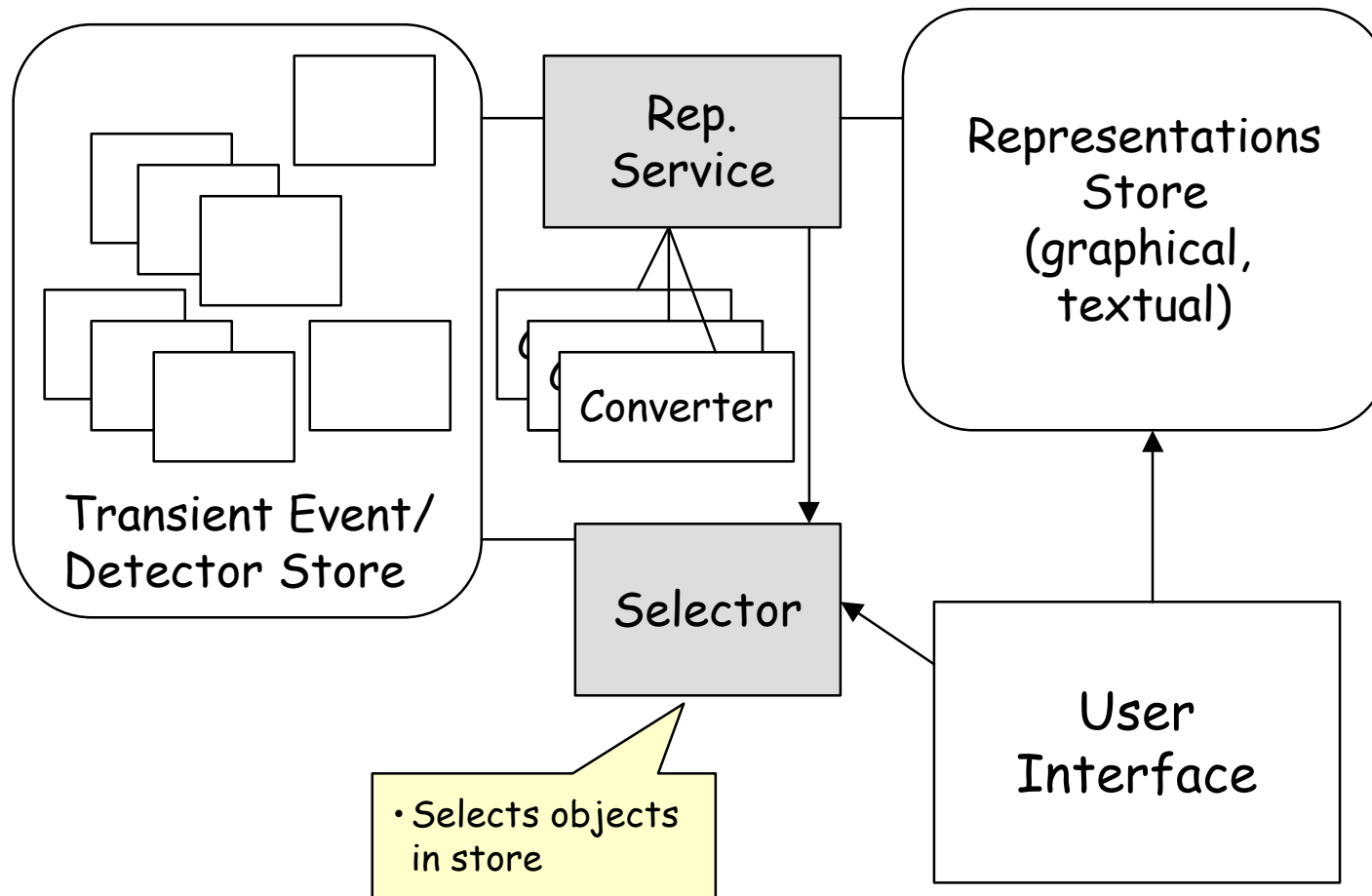
Transient/Persistent Data representations



Detector Data representations



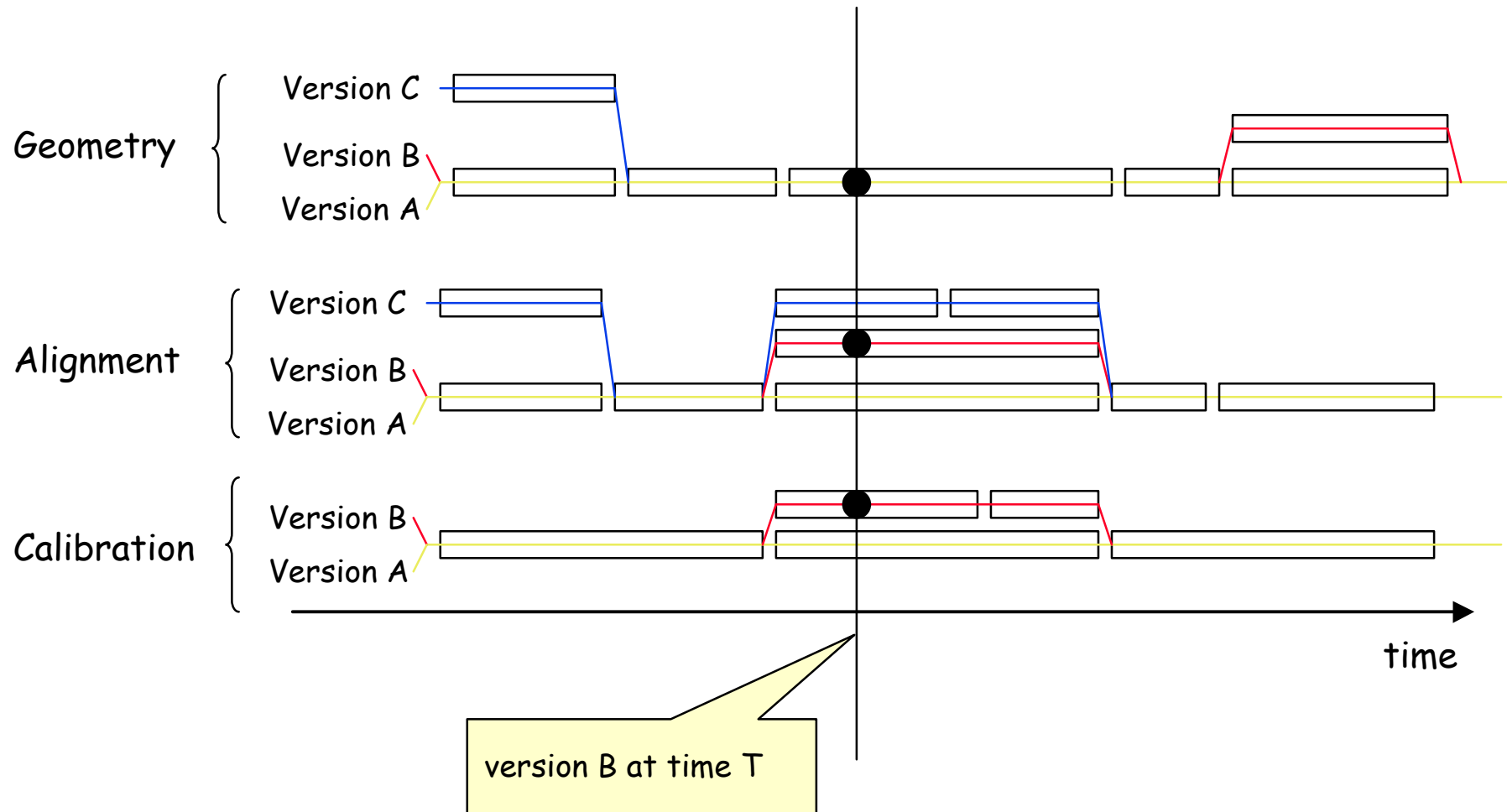
Detector description: Visualization



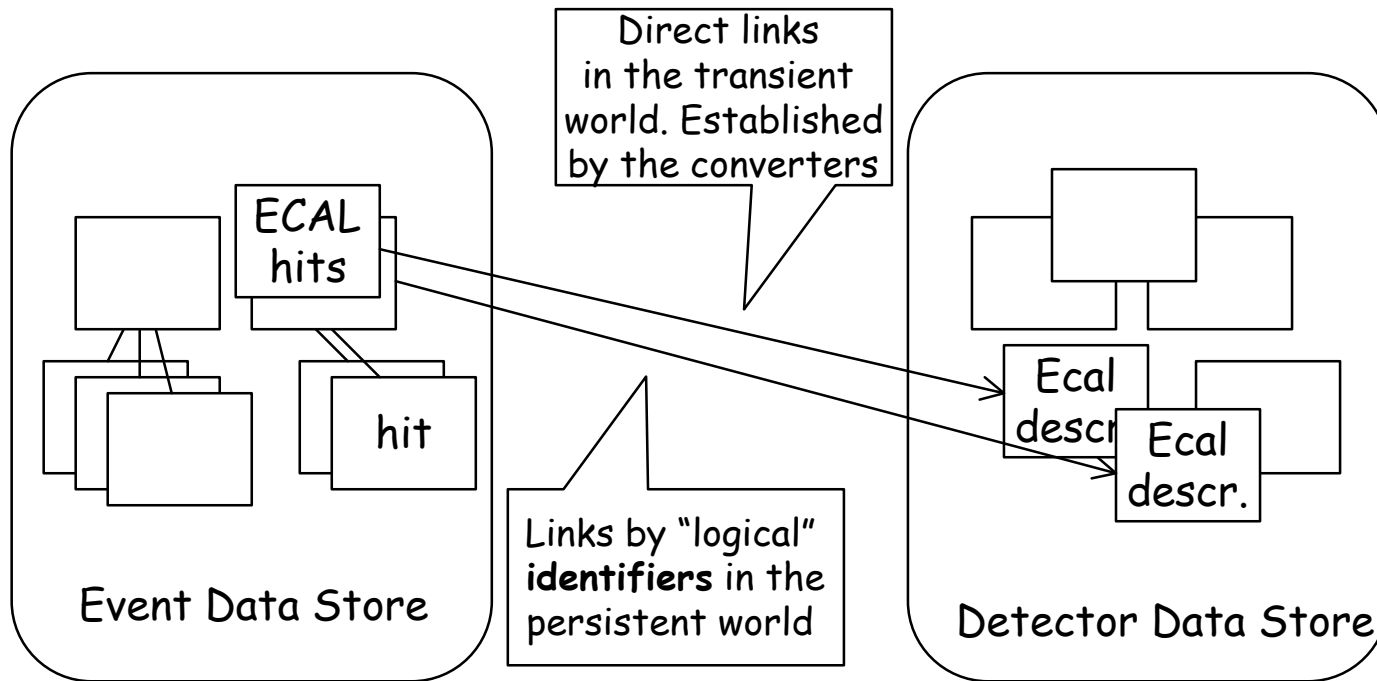
Detector Description

- ◆ 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 process and a labeled version.

Detector Database



Links between Event/Detector

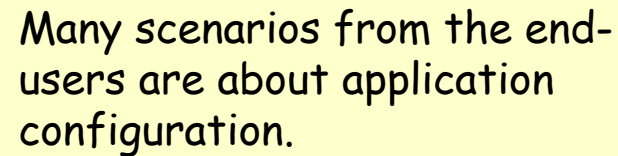


- A priori different “persistent stores”. Logical identification needed.

Analysis of Scenarios

Classification of Scenarios

- ◆ Physicists-Users
 - Job configuration
 - Algorithm configuration
 - User interactivity
 - Use of data
- ◆ Physicists-Developers
- ◆ Data production managers
- ◆ Framework developers
 - Software changes
 - Environment changes
- ◆ System maintainers



Many scenarios from the end-users are about application configuration.

Input from the reviewers

- ◆ New scenario by Thomas:

“A physicist would like to access only a part of the data of an event, e.g. only the VX clusters cluster information. The architecture should allow to access this information as fast as if it would be stored in a separate file. There should be no need to create/define ntuples” (THOMAS-1)
- ◆ Some of the questions from Vincenzo:

“What exactly differentiates “top-level” Algorithms from “nested” ones?”

“Is a jet-finder a top level and track- and cluster-finder nested?”

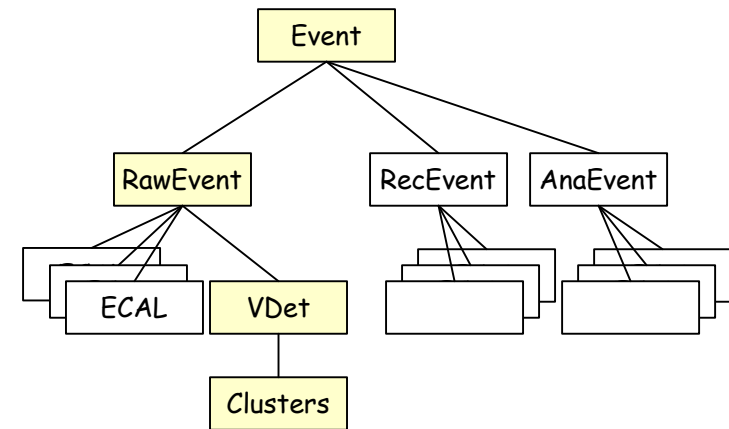
“Can two top-level algorithms share the same nested algorithms?”

“How, in this case, the cascading notification will work?”
- ◆ Ordering of interest by RD:

S-PD-9, S-PD-10, S-PD-1, S-DPM-2, S-DPM-1, S-PU-11, S-PU-14, S-PU-15
- ◆ Question about “validity of relationships” in the transient data by RD.

THOMAS-1 scenario

- ◆ Difficult to assess performance at the architecture level.
- ◆ In this case we can say that the performance **will be worst** than reading from a separate file containing the selected information. It **should be better** than reading the whole event each time since the architecture do not require it.
- ◆ How this is done?
 - The user requests:
“/event/rawevent/vdet/clusters”
 - Only the absolutely needed objects will be loaded from persistent storage.
 - Event, RawEvent, ... are small objects.



Questions from Vincenzo

- ◆ “What exactly differentiates “top-level” Algorithms from “nested” ones?”
 - Nothing. The top-level is one that is called directly by the *ApplicationMgr*
- ◆ “Is a jet-finder a top level and track- and cluster-finder nested?”
 - Yes and no.
- ◆ “Can two top-level algorithms share the same nested algorithms?”
 - Not the same instance. Since it would produce the output. Different instances yes.
- ◆ “How, in this case, the cascading notification will work?”
 - We can instantiate one than one copy of an Algorithm. The default set of properties are obtained from the property database. Each instance is identified by a name.
All the algorithms form a true tree, therefore there is no problem in cascading the notifications.

S-PD-9 scenario

“The definition of an object must be changed. The developer would like to be able to read data containing old versions of the object and data containing new versions of the object together”.

There are several cases for this scenario:

- (a) If the new version do not add more information (change of format, precision, etc.). Then the *converter* can take the responsibility of producing identical transient representations for both versions. The *Algorithm* will be affected at all.
- (b) If the old version is lacking information (items being added). Then the converter can not invent the missing information and some data fields will be left blank. The Algorithm will need to be coded to take this eventuality into account.

S-PD-10 scenario

“In order to optimize the geometry of a sub detector a physicist wishes to run the event simulation on two different geometries and compare the results. For some aspects an event by event comparison would be useful.”

There are several possibilities. One of them is:

- Two jobs are setup (configured) to use two versions of the geometry database.
- The simulation output is put into two different “directories” for each event.
- A third job is run to compare the two simulations event by event

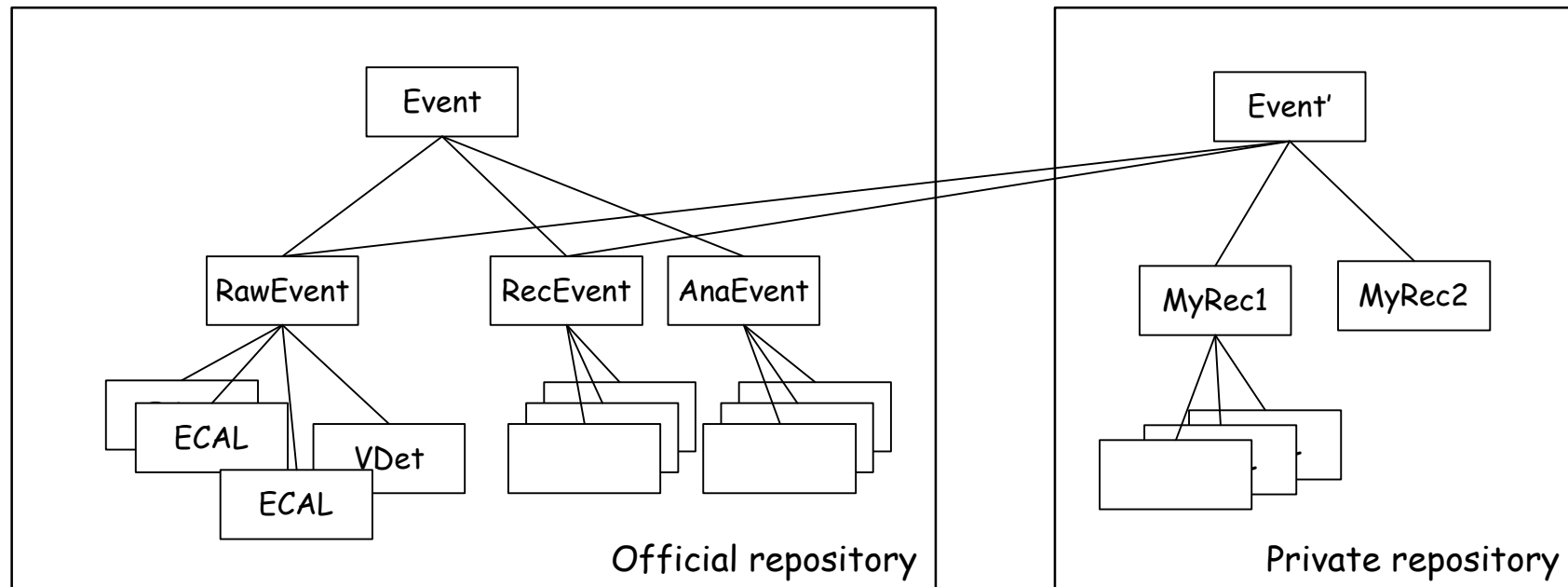
Another option is:

- Create two instances of the *transient detector store* configured to take a different geometry version each one.
- Also create two algorithms which simulate the detector and put the results in two different “directories”.
- Finally, a third algorithm to compare the results.

S-PD-1 scenario

“There is a central database of generated data. Two independent reconstruction developers read this data and generate their own data types. Both wish to save their objects along with references to the objects in the original database.”

For new privately event data we foresee to have a mechanism that allows the user to have access to his data and links to the constituent data without modifying it.



S-PU-9 scenario

“Two independent detector alignments are available. A physicist wishes to compare them by measuring the difference in the fitted momentum of individual tracks.”

- Assuming that the transient detector store has **only one version** of the detector data (corresponding to a version name and the time of the current event)
- We need to create two *transient detector stores* with the corresponding services. Should be possible.
- There should be one algorithm which creates two copies of the fitting algorithm:
 - » each one attached to a different *DetectorDataSvc*
 - » each one outputting the fitted tracks in a different “directory” in the unique *transient event store*.
- After the tracks are fitted, the algorithm can compare them and produce its results.

S-PU-10 scenario

“A track-fit algorithm is executed on a set of hits and a track produced. The algorithm is then re-run with parameters modified according to the momentum of the fitted track and a new track produced.”

- This is just a single algorithm which requires two passes.
- The first track which is produced is temporary and it only serves to obtain the momentum. It is deleted afterwards.
- The second pass is called with the updated information.
- The results are registered in the transient event store.
- It could be implemented as one parent algorithm which calls in two passes its track-fit algorithm

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.