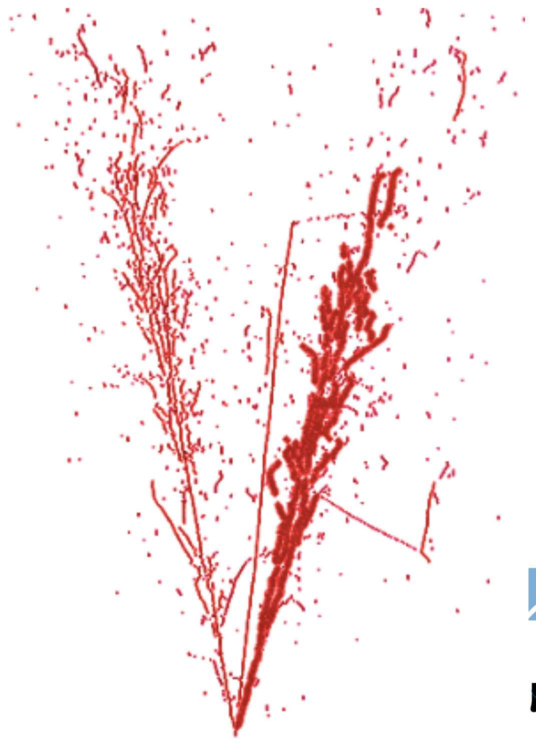
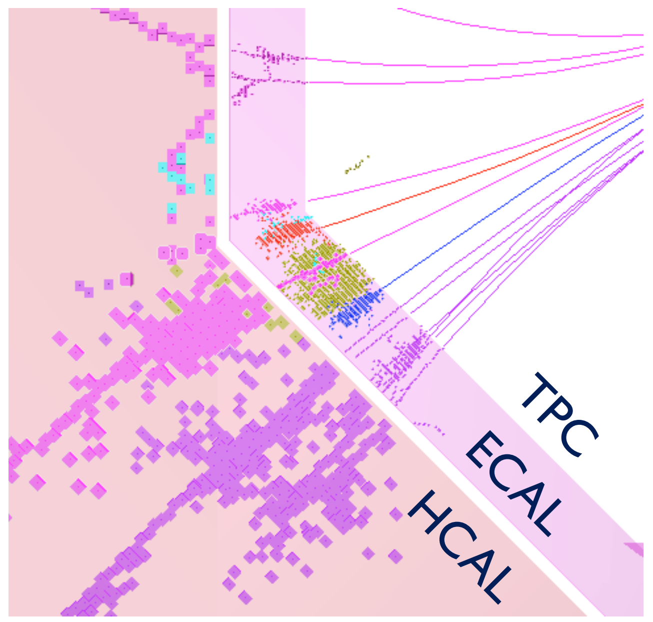


Pandora PFA v02-01-00

J. S. Marshall, A. S. T. Blake, M. A. Thomson

8 September 2015

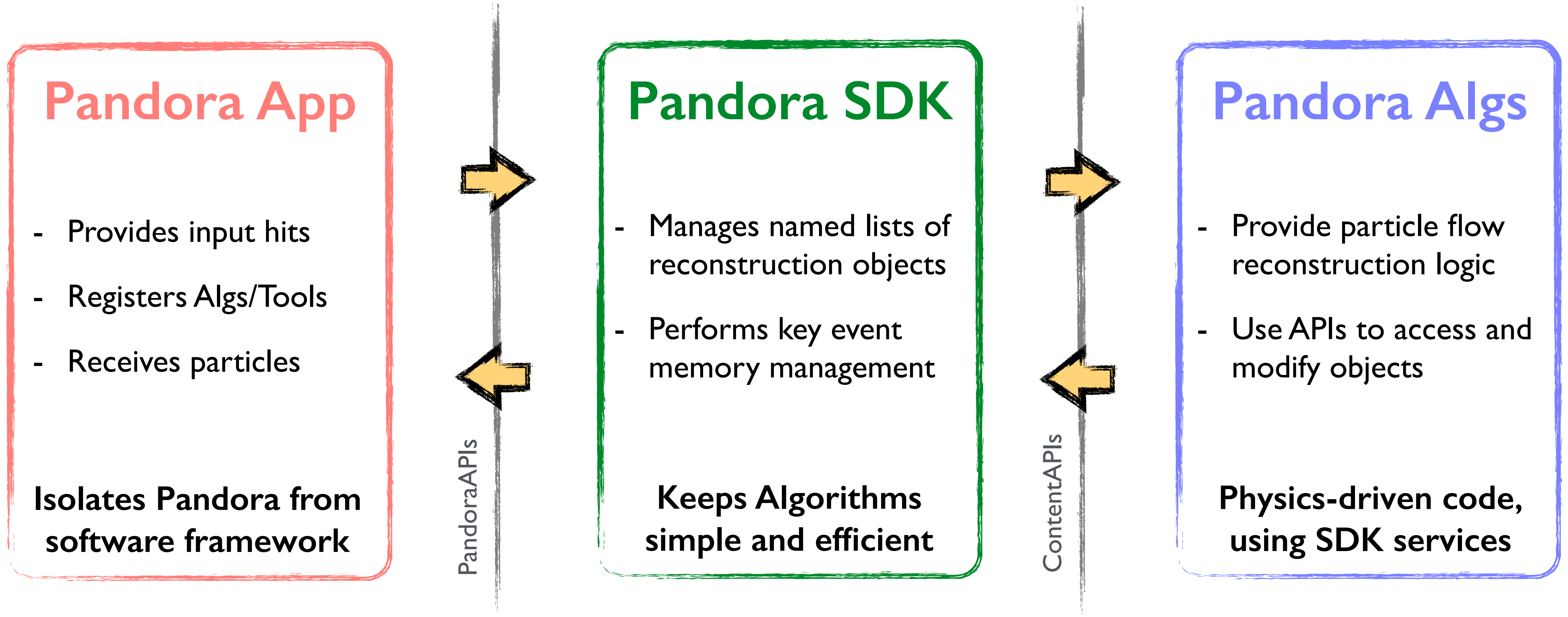


Pandora Approach

The Pandora Software Development Kit provides a software environment in which:

1. It is easy for users to provide the building-blocks that define a pattern recognition problem.
2. Logic required to solve pattern recognition problems is cleanly implemented in algorithms.
3. Operations to access or modify building-blocks, or to create new structures, are requested by algorithms and performed by the Pandora framework.

This promotes the use of large numbers of algorithms, each addressing specific event topologies.



Pandora App

- Provides input hits
- Registers Algs/Tools
- Receives particles

Isolates Pandora from software framework

Pandora SDK

- Manages named lists of reconstruction objects
- Performs key event memory management

Keeps Algorithms simple and efficient

Pandora Algs

- Provide particle flow reconstruction logic
- Use APIs to access and modify objects

Physics-driven code, using SDK services

http://github.com/PandoraPFA/

https://svnsrv.desy.de/viewvc/PandoraPFANew/

Pandora Particle Flow Algorithm v02-01-00

• Pandora PFA v02-01-00	⟵ Metadata/build package, collects consistent tags
• Pandora SDK v02-00-01	⟵ Pandora framework
• Pandora Monitoring v02-00-00	⟵ ROOT EVE visualisation, plus tree writing
• Pandora LArContent v02-01-00	⟵ Algorithms for use with a single-phase LAr TPC
• LArPandora	⟵ Client App, an art producer

- Branch "feature/blake_prepareForPandoraV2" pushed to the fermilab remote
- Latest Pandora steering files (algorithm choice/config) in LArPandoraInterface/scripts

Previous version of Pandora PFA distributed as a LArSoft external was v01-01-00.

Changes to the daughter libraries are described in the following slides…

Pandora SDK Changes

- **Major change:** Implementation of custom object creation factories, with plugins for Pandora persistency. User can modify object construction parameters and control object allocation.
- **Associated:** Reduce previous Pointing and Rectangular CaloHits to single CaloHit class.
- **Associated:** Interface changes for persistency.

- Can now extend objects in Pandora EDM with LAr-specific content.
- Should use sparingly, but this is powerful functionality.

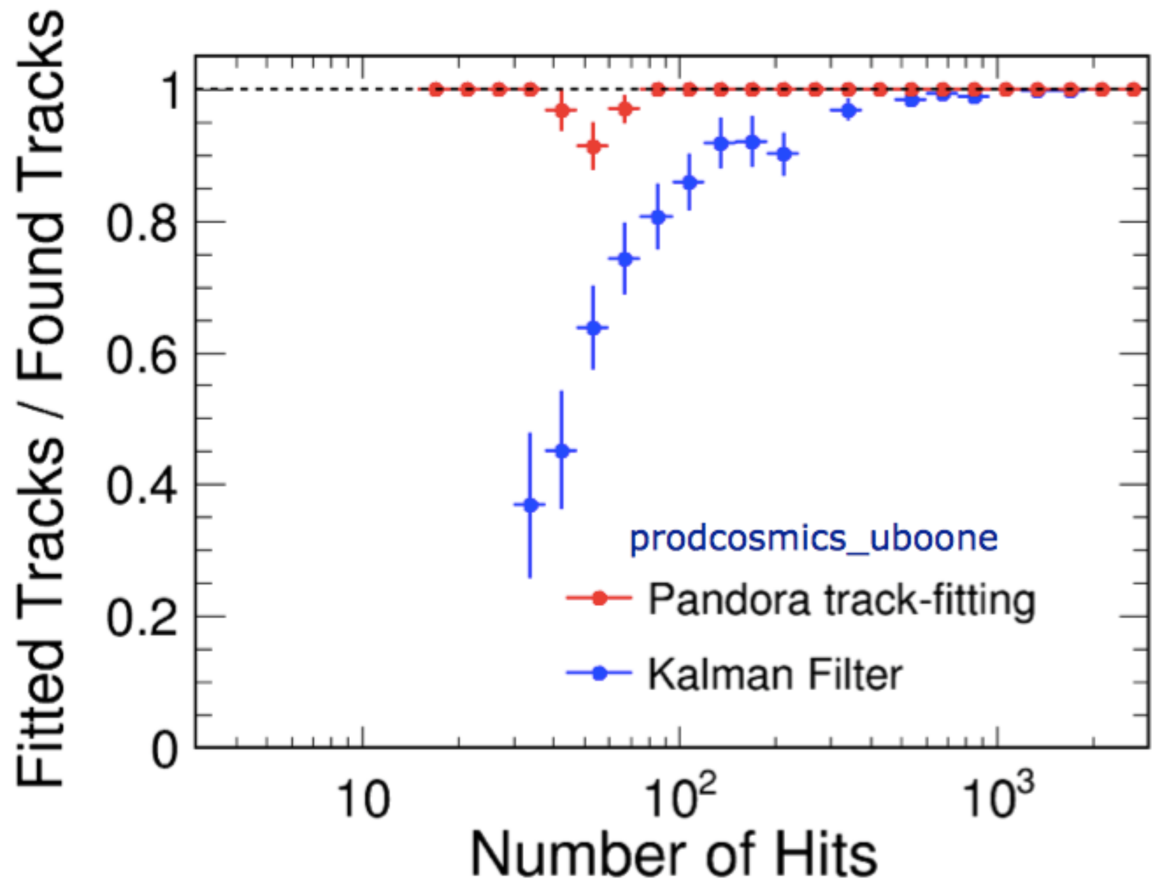
E.g. Factory allocates an "ExampleCaloHit" instance, which derives from base CaloHit in Pandora EDM.

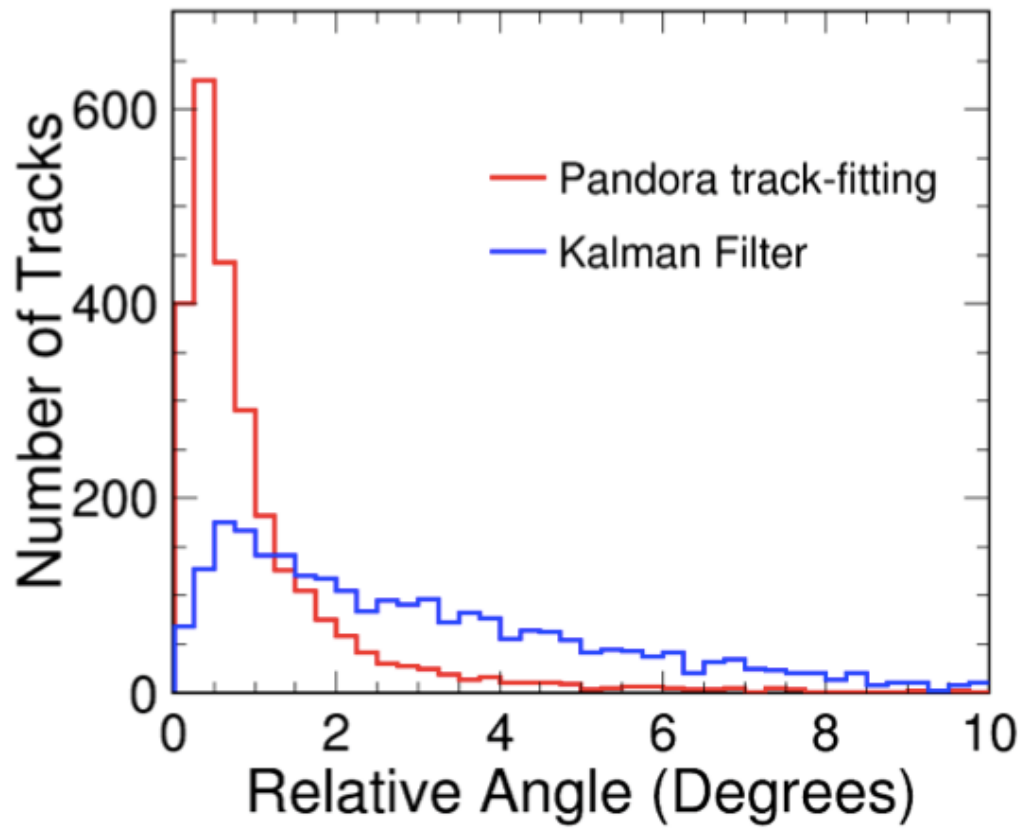
Pandora SDK works with pointers to base class, but algorithms can access additional functionality via use of a dynamic cast.

```
// In an example Pandora algorithm
ExampleCaloHitFactory exampleCaloHitFactory;

for (unsigned int iHit = 0; iHit < m_nHitsToMake; ++iHit)
{
    ExampleCaloHitParameters parameters;
    parameters.m_additionalProperty = "ExampleAdditionalProperty";
    parameters.m_positionVector = …;
    // Assign to all required properties here

    const CaloHit *pCaloHit(nullptr);
    (void) PandoraContentApi::CaloHit::Create(*this, parameters, pCaloHit, exampleCaloHitFactory);
    …
}
```

Example Usage

- Use new functionality to persist results of Pandora "fast" track-fits performed by Pandora algorithms during reconstruction.
- Client application can now access the fast fit results and build recob::Track objects from track-like particles, persisting information that was otherwise thrown away.
- Provides useful extensions to an EDM that must remain sufficiently generic to support its (multiple) Linear Collider, CMS and LAr TPC use-cases.





Pandora Additional Changes

Pandora LArContent

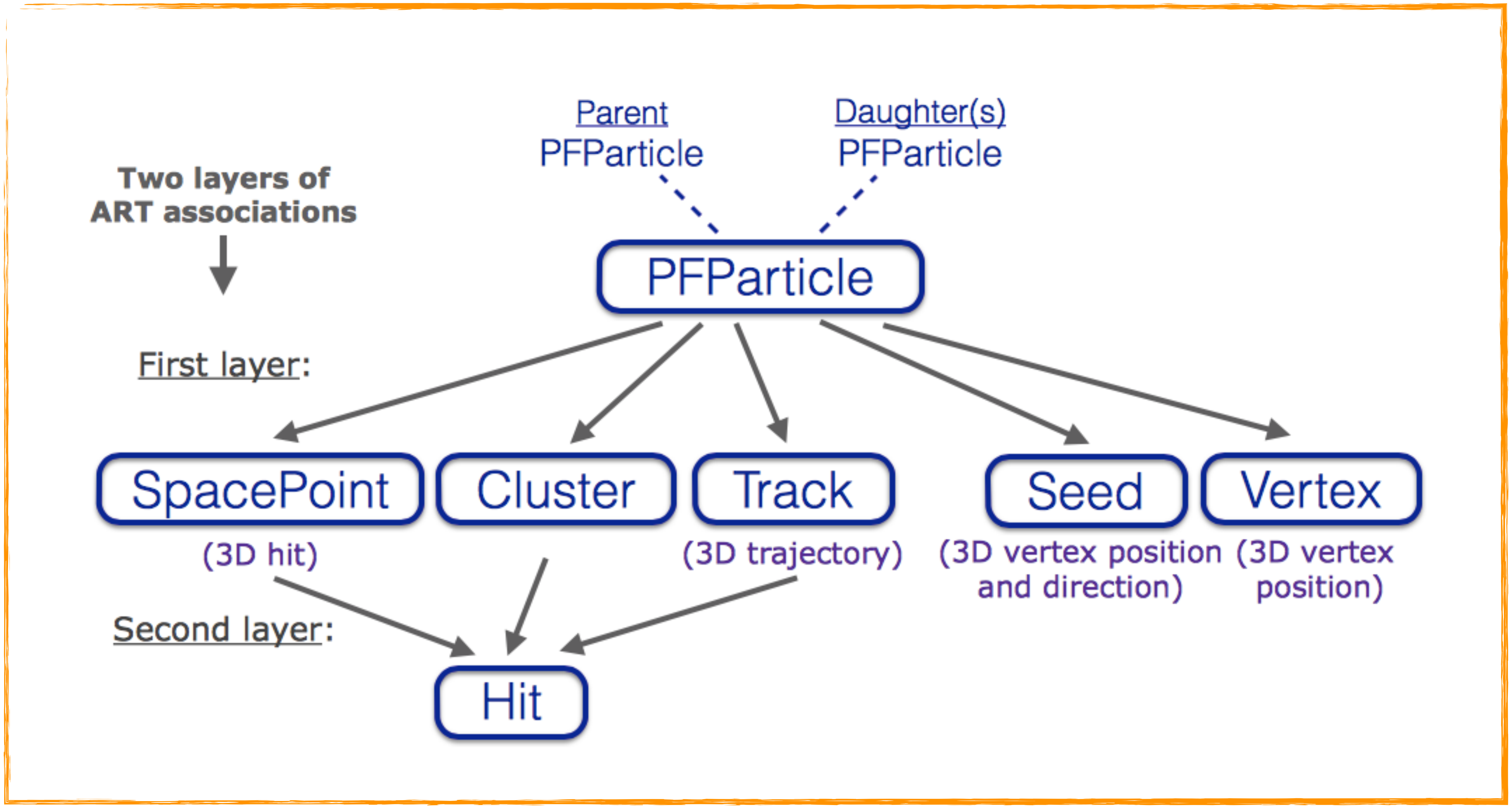
- Decorated objects in Pandora EDM: LArMCParticle, LArTrackPfo and LArShowerPfo (WiP).
- New algs and tools for fully-reconstructed neutrino interaction hierarchy (parent neutrino has reconstructed primary particles as daughters, some of which have e.g. decay daughter particles).
- New algs to improve reconstruction of short CR muons.
- New algs to pick-up residual tracks around the reconstructed neutrino interaction vertex.
- New validation algs, which carefully match reconstructed particles to primary MCParticles.
- Algs that use MCParticles to "cheat" aspects of pattern-recognition (for development use only!)

Pandora Monitoring

- Minor changes, required only to address a small breaking change in SDK.

Additional details in distributed ChangeLogs

Pandora Output to LArSoft



Pandora Notes on current distribution

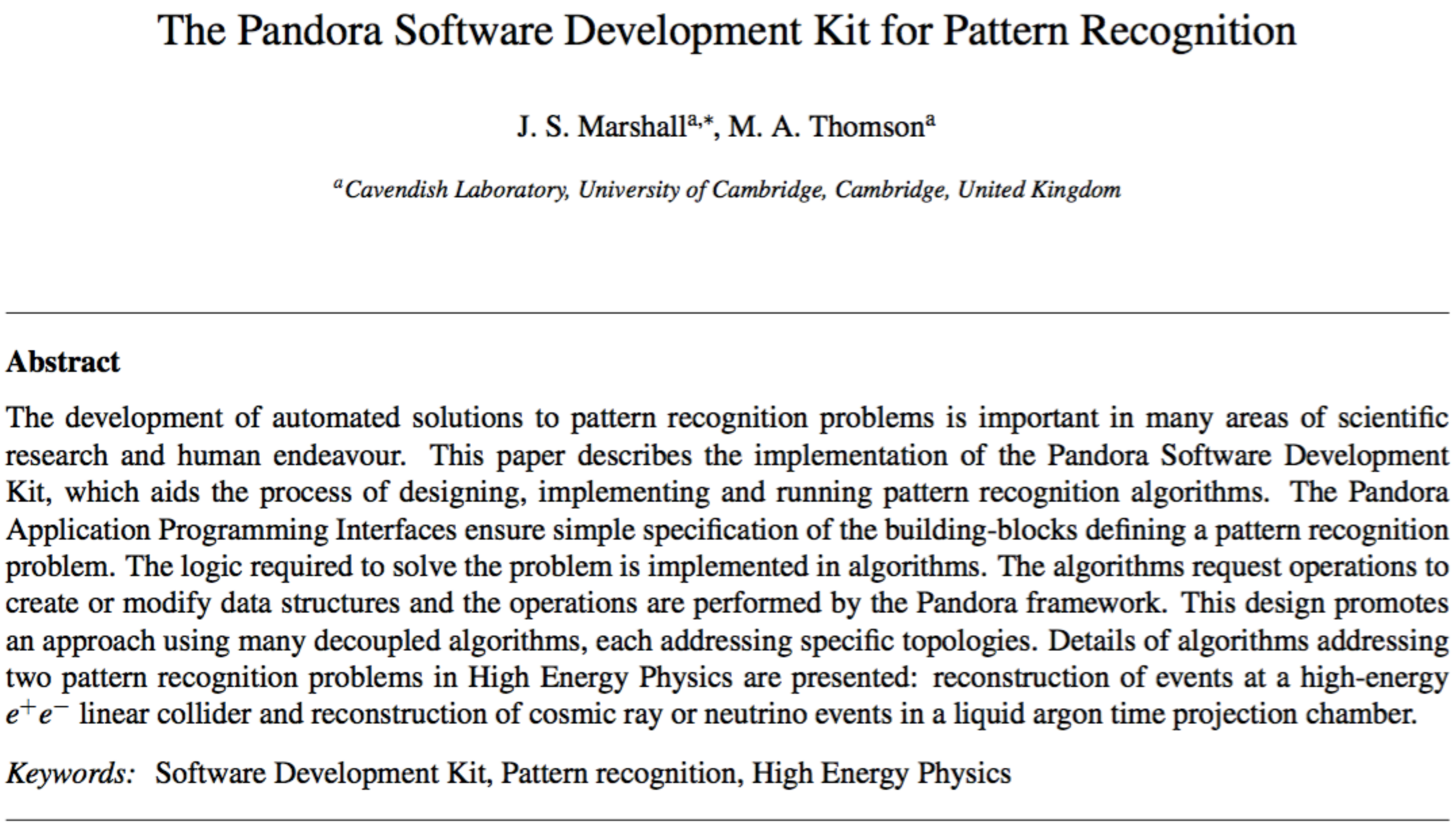
- Unorthodox to distribute the actual reconstruction algorithms as externals! Some concern about increased time taken for users to access new developments.
- May also inhibit new algorithm contributions (although already steep learning curve - need to be keen on our multi-algorithm approach to event reconstruction to get involved).
- Developments only made available when we reach particular milestones, so end-users see steps (hopefully in right direction!) in performance of the black-box Pandora reconstruction.
- Overhead associated with fixing any small (e.g. very rare) errors we don't catch. In worst-case, last-minute scenario can "promote" individual algorithms into Client App and override external.

- Original idea was to have LArContent as a separate git repository (as e.g. see our github). Difficulties encountered with having client app and LArContent library in same repository.
- Client app unfortunately has dependency on some helper functions in LArContent, not just SDK. Exposing LArContent apparently required altering include path structure to match repository(?).
- Forces an inappropriate/difficult/invalid structure on our standalone development environment, repositories and static analysis (Coverity) builds at CERN. **Stay as we are for now?!**

Pandora A Few More Details

Pandora Documentation

http://arxiv.org/abs/1506.05348



The Pandora Software Development Kit for Pattern Recognition

J. S. Marshall[a,*], M. A. Thomson[a]

[a] *Cavendish Laboratory, University of Cambridge, Cambridge, United Kingdom*

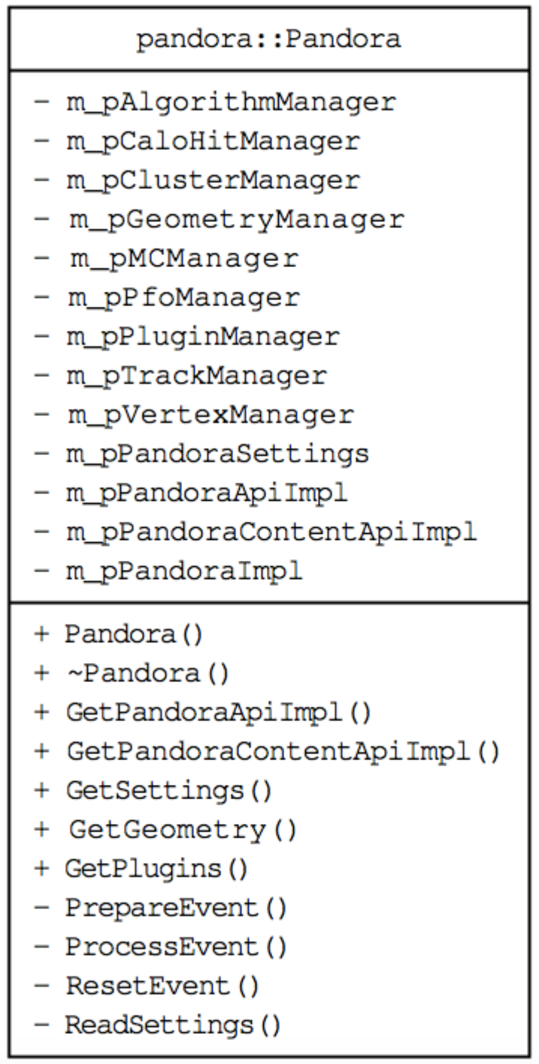
Abstract

The development of automated solutions to pattern recognition problems is important in many areas of scientific research and human endeavour. This paper describes the implementation of the Pandora Software Development Kit, which aids the process of designing, implementing and running pattern recognition algorithms. The Pandora Application Programming Interfaces ensure simple specification of the building-blocks defining a pattern recognition problem. The logic required to solve the problem is implemented in algorithms. The algorithms request operations to create or modify data structures and the operations are performed by the Pandora framework. This design promotes an approach using many decoupled algorithms, each addressing specific topologies. Details of algorithms addressing two pattern recognition problems in High Energy Physics are presented: reconstruction of events at a high-energy e^+e^- linear collider and reconstruction of cosmic ray or neutrino events in a liquid argon time projection chamber.

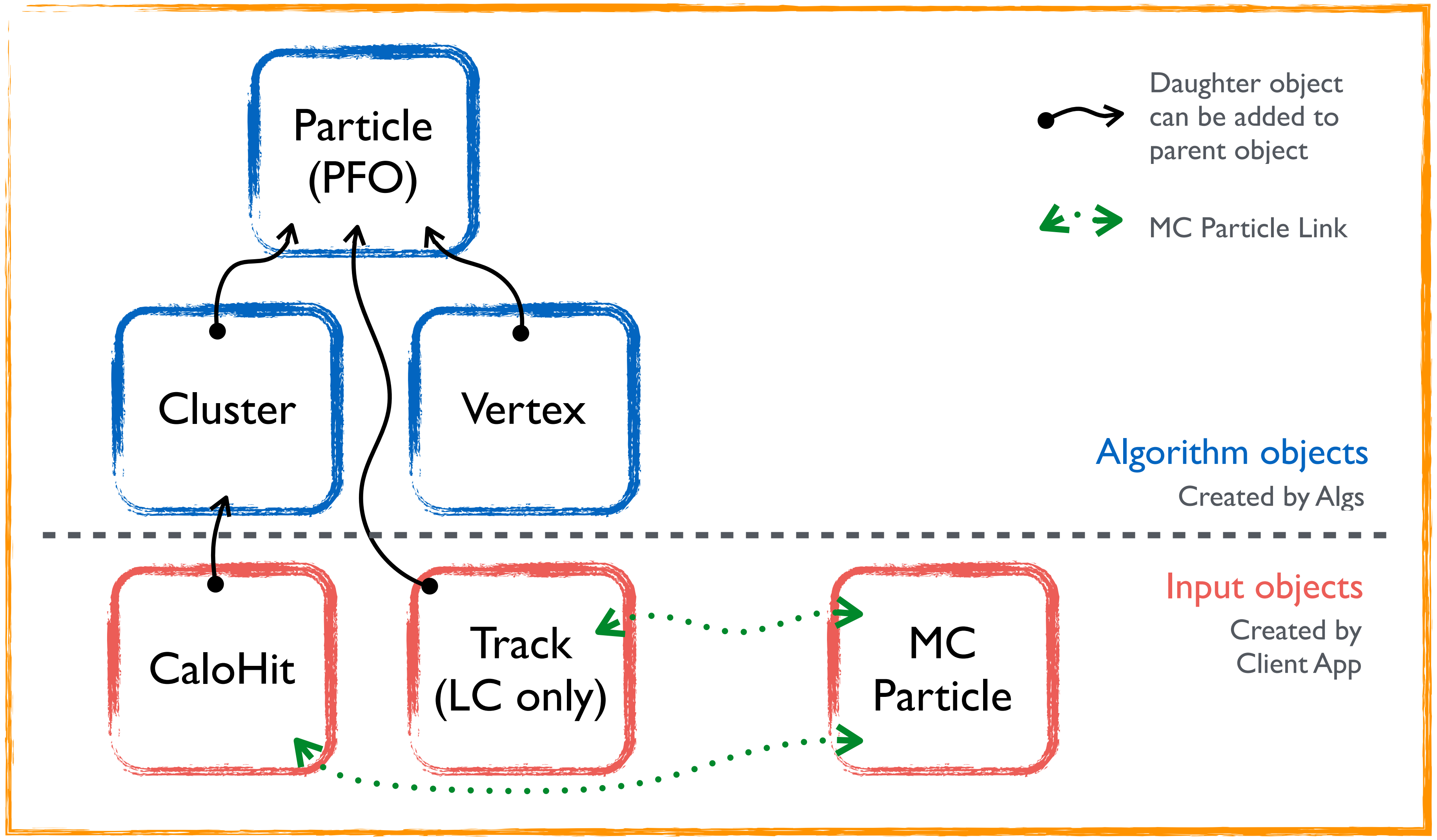
Keywords: Software Development Kit, Pattern recognition, High Energy Physics

Pandora SDK

- The Pandora SDK provides a comprehensive **Event Data Model** (EDM) for managing pattern recognition problems. Instances of objects in the EDM are owned by **Pandora Managers**.
- The object instances are stored in named lists and the Managers are able to create new objects, delete objects, create and save new lists and move objects between lists.
- The Managers provide a complete set of low-level operations that allow the high-level operations requested by pattern recognition algorithms to be satisfied.

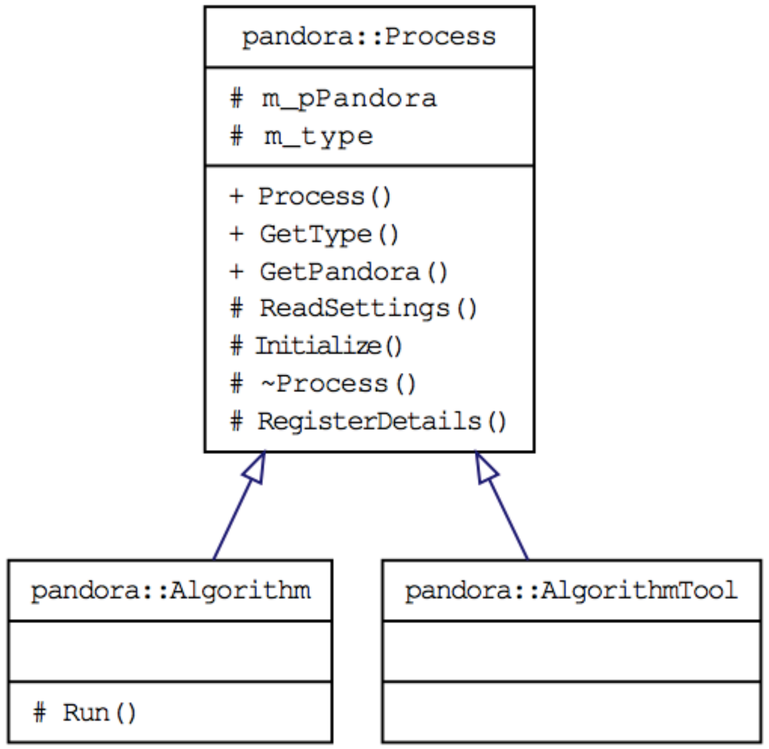


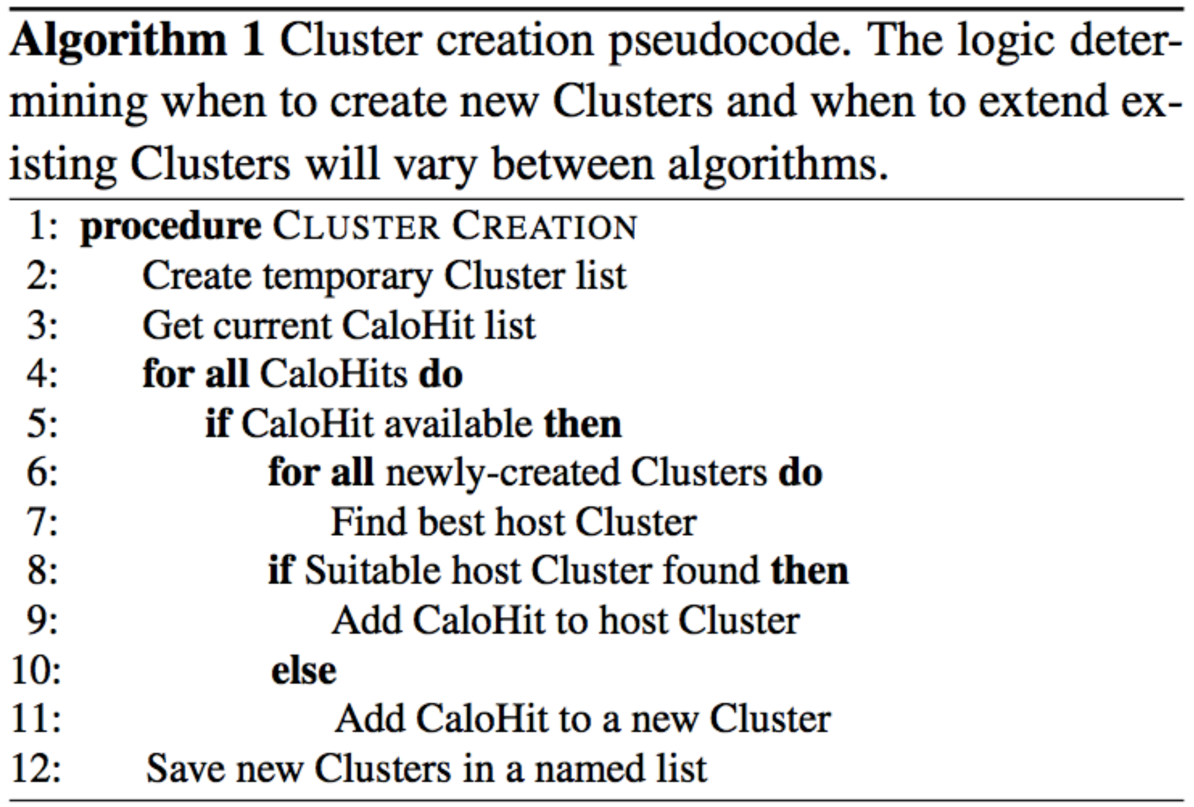
```
pandora::Pandora
- m_pAlgorithmManager
- m_pCaloHitManager
- m_pClusterManager
- m_pGeometryManager
- m_pMCManager
- m_pPfoManager
- m_pPluginManager
- m_pTrackManager
- m_pVertexManager
- m_pPandoraSettings
- m_pPandoraApiImpl
- m_pPandoraContentApiImpl
- m_pPandoraImpl
+ Pandora()
+ ~Pandora()
+ GetPandoraApiImpl()
+ GetPandoraContentApiImpl()
+ GetSettings()
+ GetGeometry()
+ GetPlugins()
- PrepareEvent()
- ProcessEvent()
- ResetEvent()
- ReadSettings()
```



Pandora Algorithms

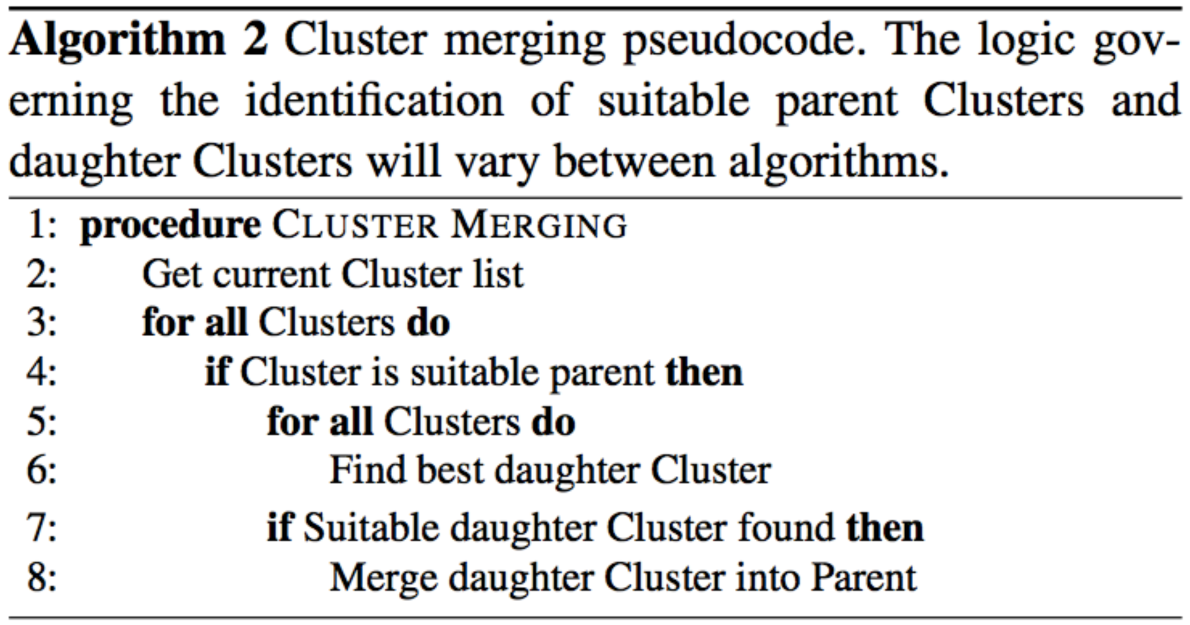
- Pandora algorithms contain the step-by-step instructions for finding patterns in the provided data.
- They use the APIs to access objects and to request the Managers to make new objects or modify existing objects.
- They inherit from the **Process** class, which provides functionality for handshaking with Pandora, XML config and function callbacks.





Algorithm 1 Cluster creation pseudocode. The logic determining when to create new Clusters and when to extend existing Clusters will vary between algorithms.

```
1:  procedure CLUSTER CREATION
2:      Create temporary Cluster list
3:      Get current CaloHit list
4:      for all CaloHits do
5:          if CaloHit available then
6:              for all newly-created Clusters do
7:                  Find best host Cluster
8:              if Suitable host Cluster found then
9:                  Add CaloHit to host Cluster
10:             else
11:                 Add CaloHit to a new Cluster
12:     Save new Clusters in a named list
```



Algorithm 2 Cluster merging pseudocode. The logic governing the identification of suitable parent Clusters and daughter Clusters will vary between algorithms.

```
1:  procedure CLUSTER MERGING
2:      Get current Cluster list
3:      for all Clusters do
4:          if Cluster is suitable parent then
5:              for all Clusters do
6:                  Find best daughter Cluster
7:              if Suitable daughter Cluster found then
8:                  Merge daughter Cluster into Parent
```

Pandora Client App

- To use the Pandora SDK, a user must create a Pandora client application. This provides the input building-blocks to describe the pattern recognition problem and receives the final output.
- The client application is responsible for controlling the pattern recognition reconstruction. It creates the Pandora instance(s) and uses Pandora APIs to send requests to these instances.

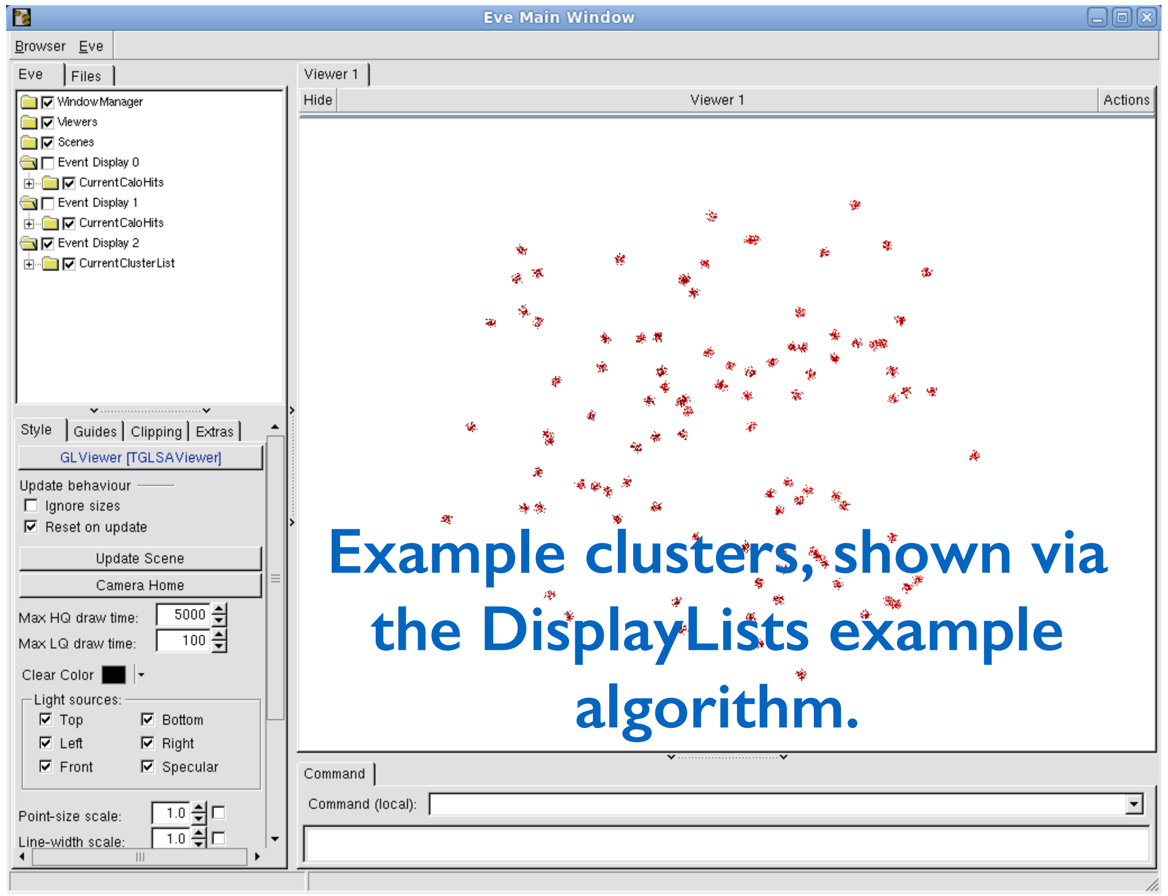
- For LArSoft, the Pandora client application is **LArPandoraInterface**
- All Algorithms and Plugins are built as part of the LArContent library.
- Pandora Hits (and, optionally, MCParticles), are extracted from named art producers.
- The output consists of collections of PFParticles, Clusters, Tracks, Showers, Vertices

Algorithm 3 Pseudocode description of a client application for LAr TPC event reconstruction in a single drift volume

```
1:  procedure MAIN
2:      Create a Pandora instance
3:      Register Algorithms and Plugins
4:      Ask Pandora to parse XML settings file
5:      for all Events do
6:          Create CaloHit instances
7:          Create MCParticle instances
8:          Specify MCParticle-CaloHit relationships
9:          Ask Pandora to process the event
10:         Get output PFOs and write to file
11:         Reset Pandora before next event
```

Pandora Learning Library

- Pandora algorithms create and/or modify clusters, vertices and PFOs. Their decisions (the algorithm logic) whether to proceed with operations can be complex and use-case specific.
- The aim of the Pandora ExampleContent library and test application is to demonstrate the key Pandora functionality in a very simple testing and learning environment.
- The ExampleContent library is structured in exactly the same manner as the LCContent and LArContent libraries, currently in use for Linear Collider and LAr TPC reconstruction.



Example clusters, shown via the DisplayLists example algorithm.

- The library consists of example Algorithms, AlgorithmTools, Plugins and Helper functions:
  - Example list access and display
  - Example Cluster, Vertex and PFO creation
  - Cluster manipulation, including merging, deletion, fragmentation and reclustering
  - Creating and saving new lists of objects
  - Using Algorithm Tools and Plugins
  - Writing a tree using PandoraMonitoring.

http://www.hep.phy.cam.ac.uk/~marshall/PandoraExample.pdf