Architecture TaskForce

Stephen Haywood (ATLAS/RAL)

ATF

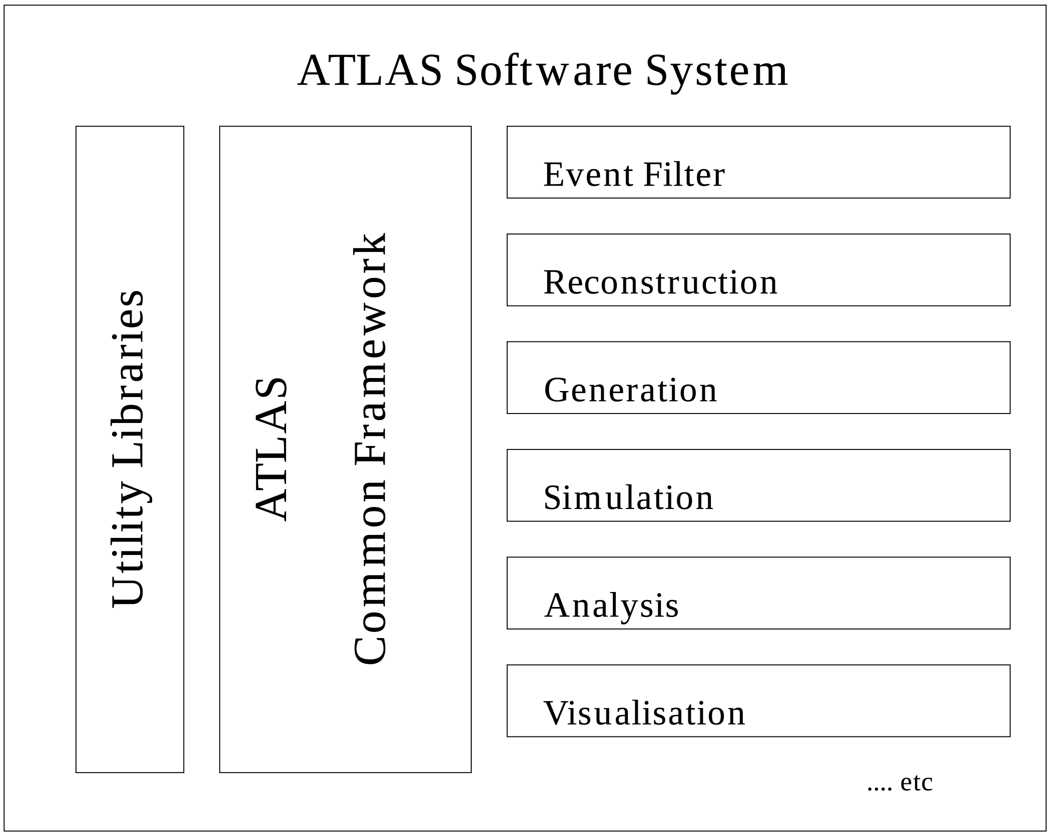
• Katsuya Amako (KEK)	GEANT4
• Andrea Dell'Acqua (CERN)	Simulation
• Helge Meinhard (CERN)	Former DIG
• RD Schaffer (CERN+LAL)	Database
• David Quarrie (LBNL)	BaBar
• Marjorie Shapiro (LBNL)	CDF
• Laurent Chevalier (Saclay)	Muons, F77 code
• Fabiola Gianotti (CERN)	Physics Coordinator
• Stephen Haywood (RAL)	Chair
• Norman McCubbin (RAL)	Computing Coordinator
• Valerio Vercesi (INFN)	Event Filter
• Torsten Akesson (CERN)	ATLAS Management

Other Inputs

ATLAS, LHCb (Gaudi), D0, ALICE (AliRoot)

Global Architecture

→ Unified Execution Framework



Emphasis on common aspects of software

... not complete architecture for all applications - follows later and is responsibility of existing groups (subdetectors and applications)

An ATLAS Design

ATF wanted to consider an “ab initio” design:

- Have the time.
- Should understand what ATLAS wants - may differ from others.

Lucky to have Katsuya Amako with us for 3 months.

Agree to follow broadly approach used by GEANT4 - along lines of Unified Software Development Process (USDP).

Requirements $\rightarrow$ Analysis $\rightarrow$ Design $\rightarrow$ Implementation $\rightarrow$ Testing

Determine what Users want $\rightarrow$ Use-cases. These

- Help identify common aspects of S/w.
- Provide criteria with which S/w can be judged.

In following USDP, we appreciate:

- Process is iterative and incremental.
- Pragmatic since can and should lead rapidly to first prototypes.
- Should adapt USDP according to ATLAS’s needs.

Use-cases

From user’s input, identify

- Actors - people or programs fulfilling tasks
- “Use-cases” - what they need to do

Table 0-1 Matrix of which Actors use which Use-cases.

Actors	Use-cases								
	UC-00	UC-01	UC-02	UC-03	UC-04	UC-05	UC-06	UC-07	UC-08
	(Setup + Use) ATLAS Common Framework Environment	**(Store + Retrieve) Detector Description**	**(Store + Retrieve) Magnetic Field**	**(Store + Retrieve) Event Collection**	**(Store + Retrieve) Calibration and Alignment Data**	**Share Common Reconstruction and Filter Packages**	**(Store + Retrieve) MC Event Generator Data**	**Use Histogram Packages**	**Exchange Messages**
Detector Description Provider	X	X							X
Magnetic Field Provider	X		X						X
Online Calibration and Alignment	X	X	X	X	X	X		X	X
Event Filter	X	X	X	X	X	X		X	X
Offline Calibration and Alignment	X	X	X	X	X	X		X	X
Offline Event Recon-struction	X	X	X	X	X	X		X	X
Offline Event Streaming	X			X				X	X
Physics Analysis	X	X	X	X				X	X
MC Event Generator (4-vectors)	X		X				X	X	X
Detector Simulator	X	X	X	X	X		X	X	X
Visualisation	X	X	X	X	X		X	X	X

Use-cases $\rightarrow$ Top-level Components - sets of closely coupled classes

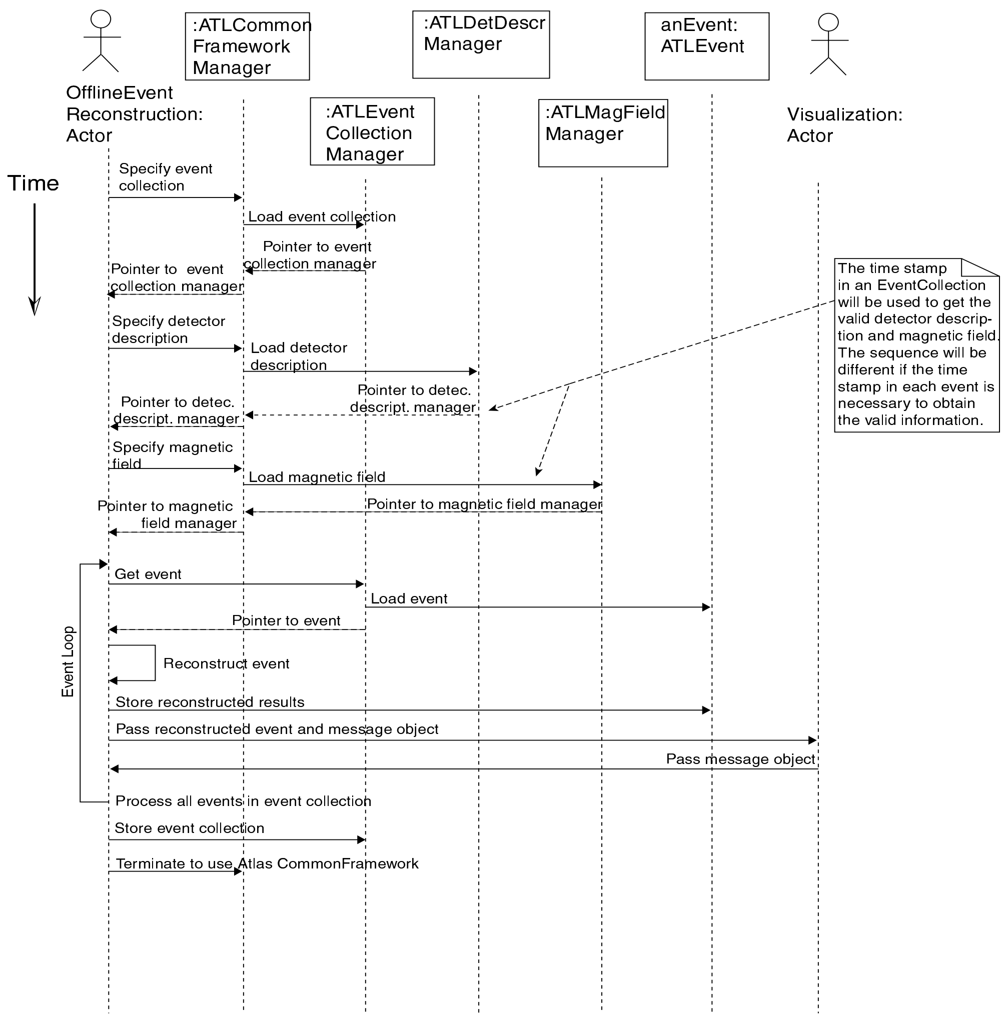
Sequence Diagram

Example: Interactive Reconstruction, including Visualisation:

Atlas Common Framework

Design Model: Sequence Diagram Ver.991018

Diagram Number: DM-SQ-00#2
Related Use Case Number: UC-00
Use Case Name : (Setup+Use) Atlas Common Framework Environment
Use Case Pattern# : UC-00#1
Description : The actor 'Offline Event Reconstruction' sets up and uses the Atlas common framework to reconstruct events. Reconstructed objects in an event are passed to the actor 'Visualization' to display it. This is a pattern of the interactive mode.l


OfflineEvent Reconstruction: Actor
:ATLCommon Framework Manager
:ATLEvent Collection Manager
:ATLDetDescr Manager
:ATLMagField Manager
anEvent: ATLEvent
Visualization: Actor
Time
Specify event collection
Load event collection
Pointer to event collection manager
Pointer to event collection manager
Specify detector description
Load detector description
Pointer to detec. descript. manager
Pointer to detec. descript. manager
Specify magnetic field
Load magnetic field
Pointer to magnetic field manager
Pointer to magnetic field manager
Get event
Load event
Pointer to event
Reconstruct event
Store reconstructed results
Pass reconstructed event and message object
Pass message object
Event Loop
Process all events in event collection
Store event collection
Terminate to use Atlas CommonFramework
The time stamp in an EventCollection will be used to get the valid detector description and magnetic field. The sequence will be different if the time stamp in each event is necessary to obtain the valid information.

- Actors
- Top-level Components (including associated classes)
- → Identification of methods - what the classes need to do

Auxiliary Reports

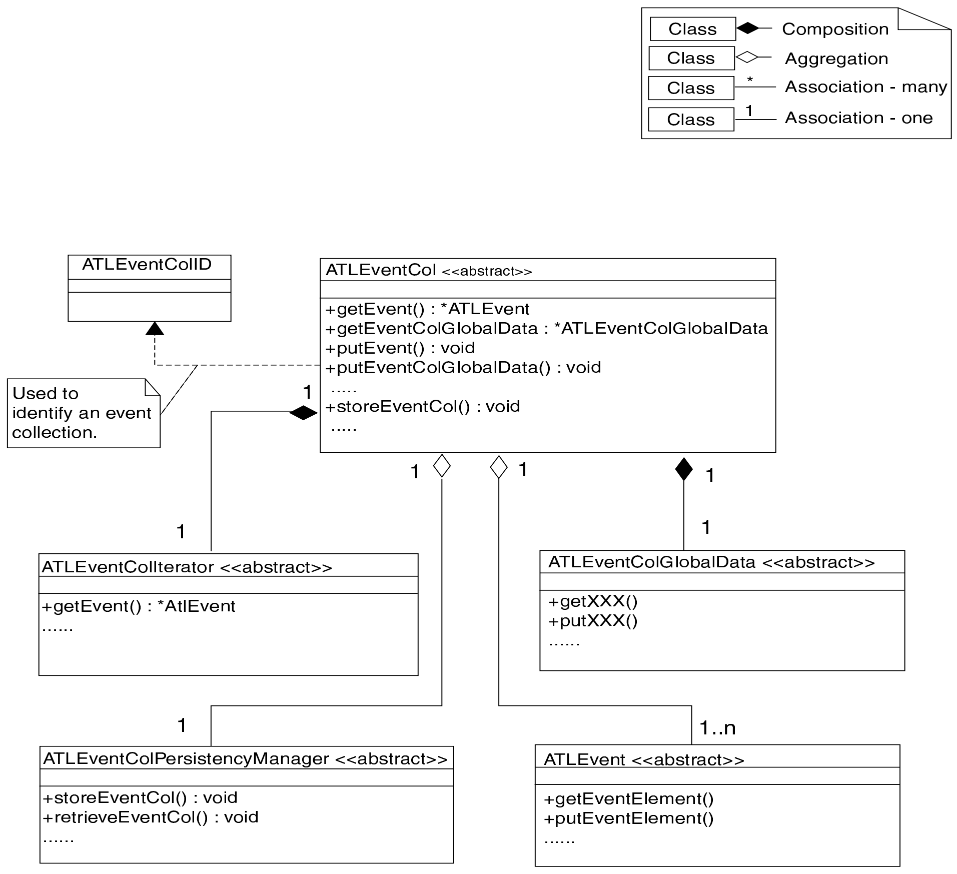
- **Use-cases**
- **First Attempt at Analysis and Design**
  Common Framework Environment
  (Store + Retrieve) Detector Description
  (Store + Retrieve) Event Collection

Example: ATLEventCol

Atlas Common Framework

Design Model: Class Diagram Ver.991018

Diagram Number: DM-CL-03#2
Related Use Case Number: UC-03
Use Case Name : (Store+Retrieve) Event Collection



- These are examples - becoming quite detailed - caveats.
- Need plenty of expert discussion.
- Lots to do.

Expected Components

Any “ab initio” approach takes time and expertise in OO approach.

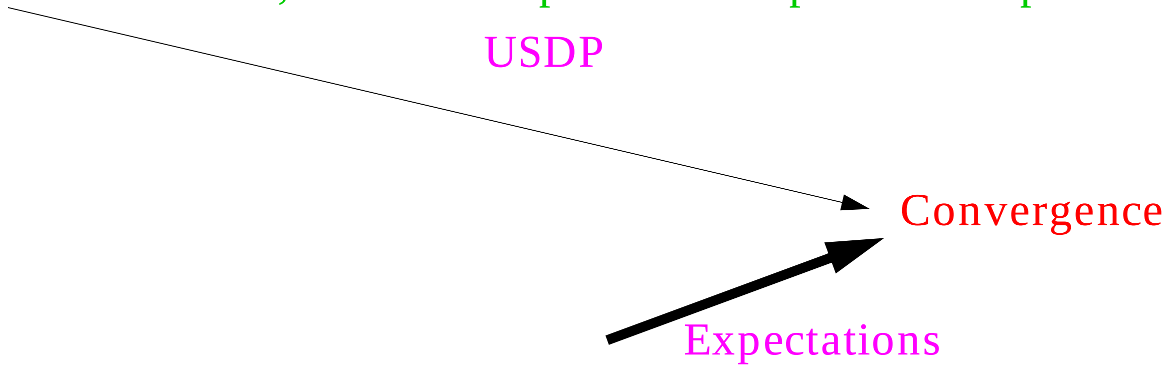
In following this, we recognise familiar concepts.
Sensible to consider:

- Our own experience, including what we have appreciated from use-cases.
- Experience of other experiments.

In Chapter 5 of our Report, we have listed the Components we would expect to see - not so much design as a compilation.

This is complementary to USDP approach

Now must develop USDP designs.
As these mature, should compare with expected components.



Convergence: Fine, USDP will provide details, i.e. responsibilities and interactions.

Non-Convergence: We have missed something in USDP or identified a different (and potentially better) design.

On-going Work

Event

Similar to Zebra Event bank, from which all banks hang.

Associated with classes which deal with

- Input/output
- Selection
- Persistent and Transient Stores

Detector Description

Very active work to develop Det Descr, involving several subsystems (using XML approach).

Database

Paso gives a very simple OO/C++ “framework” in which developers may work *today* - for example, can access Event from Zebra or Objectivity.

Decisions

OO

C++ - with an eye to Java

Separate Data and Algorithms

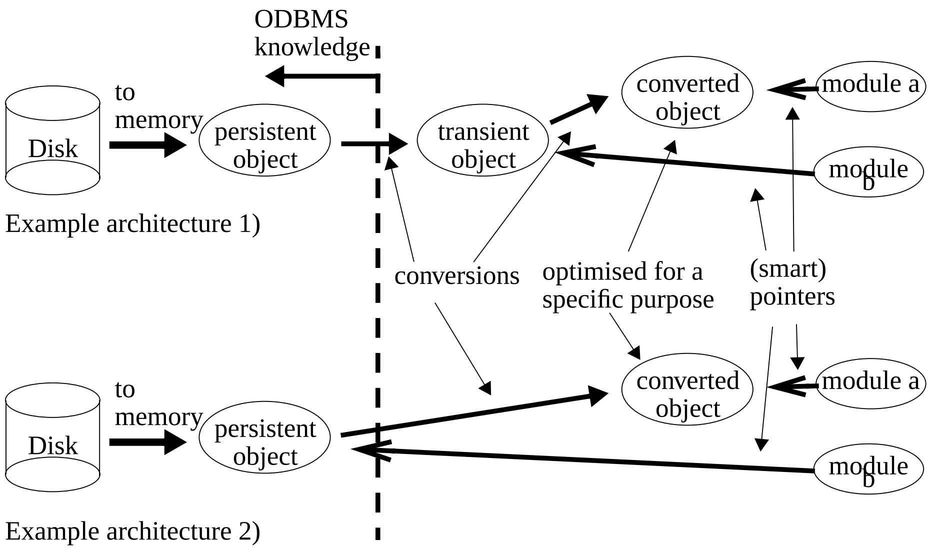
- Traditional
- A matter of degree
- Flexibility to develop Algorithms, stability of Data definitions.

On-Demand Access

Access data only when explicitly requested.

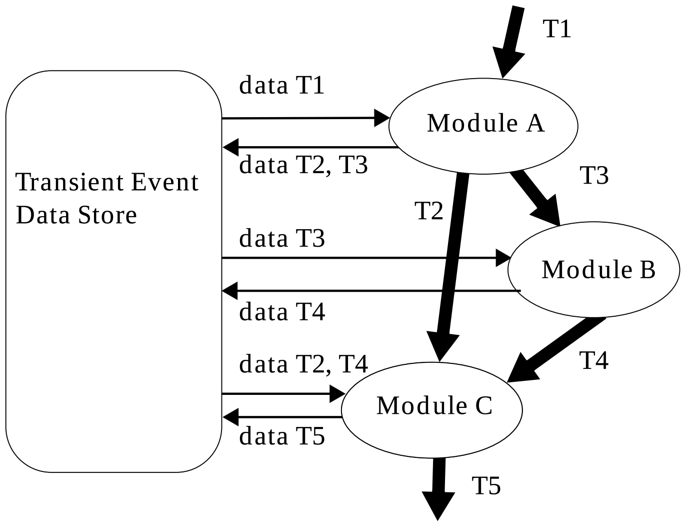
Independence from Database Supplier

- May need to consider various Databases or completely different scheme like Root Trees.
- Propose to use a Transient Store.



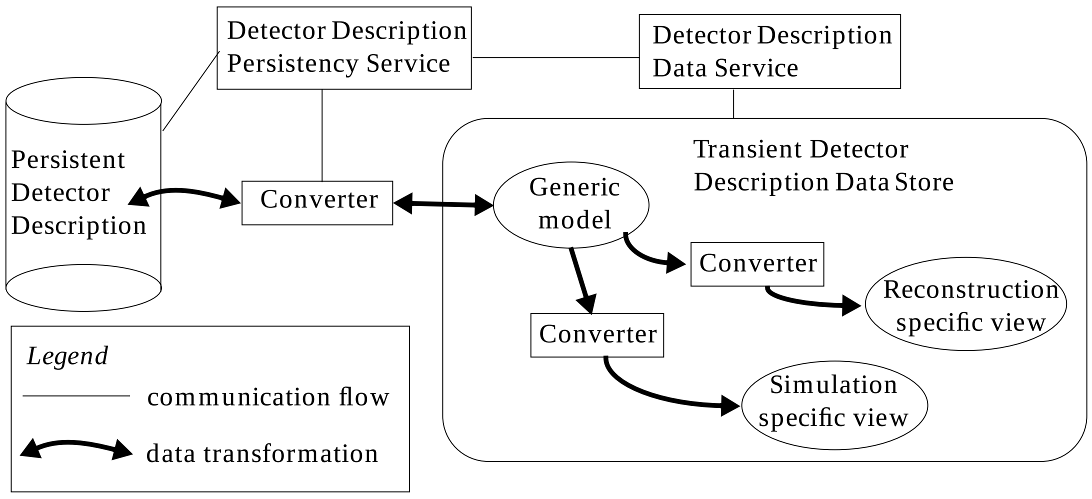
Access to Event Data via Transient Event Store

- Like traditional approach where modules read / write banks to / from Zebra memory.
- Unique source of event data; avoid ambiguities of ownership.
- Provides “blackboard” where information can be examined (subsequently select what to write to Persistent Store).



Within Modules, recommend that data is passed directly as objects.

Single Source for Detector Description


Detector Description Persistency Service
Detector Description Data Service
Persistent Detector Description
Converter
Transient Detector Description Data Store
Generic model
Converter
Reconstruction specific view
Converter
Simulation specific view
Legend
communication flow
data transformation

Control Flow

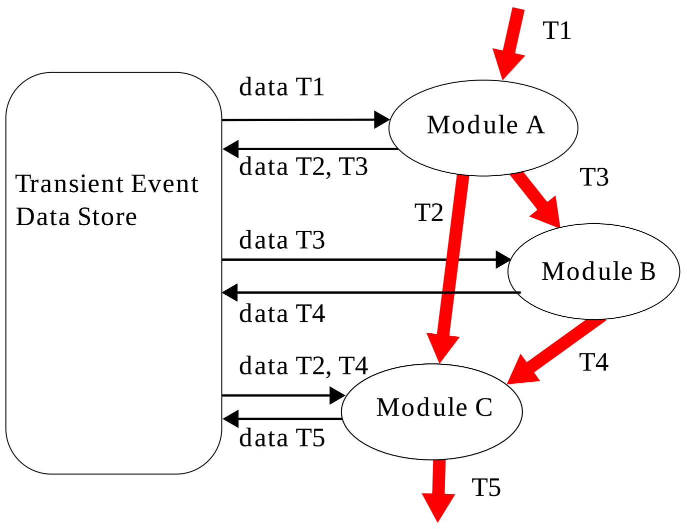
Relates to scheduling of modules ... also associated with data flow.

Traditional approach:

call A, call B, call C
- may be coded directly or prepared by a script

Object Networks (Lassi Tuura)

Scheduling of modules determined by flow of data and its availability.



Approaches may not appear to be so different logically. Differences become more apparent with more complex networks.

Issues are: Scheduling, Scalability, Filtering, Parameter Passing, Ownership, etc.

Although this is an implementation matter, it has significant architectural consequences and other decisions depend on it.

LBL Group has looked into some of the issues.

No show stoppers with ON's - they have some nice features. However there are concerns and in the absence of overwhelming advantages, ATF recommends traditional approach to control flow.

Architecture Team

- The ATF was given three months.
- We have all worked very hard - honestly ! (Esp KA, MS, DQ)
- We have done as much as a *representative committee* can do.
- Now emphasis should be on OO Design.
- Need a coherent and dedicated team.
- They must understand OO issues and work together.
- Should not be a bureaucratic committee.
- Clearly, need active involvement from the Detector and Physics Communities.

Tasks are:

- Detailed OO Design of key Components. In particular,
  - Develop the USDP Design.
  - Continue design associated with on-going work.
  - Bring the two sets of ideas together.
- Realisation of Framework - proposed prototype for ~May 2000.
- Support of Subsystems.

Subsystems

The subsystems can move forwards using: Geant4 (Chaos) and Paso.

We hope the Report contains a sufficient outline to permit progress (distinction between algorithms and data, use of Transient Data Store, module communication etc.).

(My Personal) Conclusions

- To set out the Global Architecture is a big, complicated and specialised job.
- For this reason, we did not achieve as much as I had hoped.

- The ATF has collected together for the first time in ATLAS the information related to the crucial issues for the ATLAS Architecture.
- We have identified a methodology which is capable of producing an “ab initio” design - this work must be continued.
- We have provided directions for where we expect to be heading.
- The Detector and Physics Community should be able to proceed with their own code development in anticipation of the Framework.

- We should aim for a first prototype of the Framework by May 2000 (since time is slipping, and I note the Gaudi implementation has taken more like a year, this may be too optimistic).
- We must get an Architecture Team up and running asap.