Software Components, Schedule and Manpower…

Dave Abbot and I, as CAMs are iterating on the cost/schedule. I want to discuss the software/firmware tasks that need to be divided into work packages:

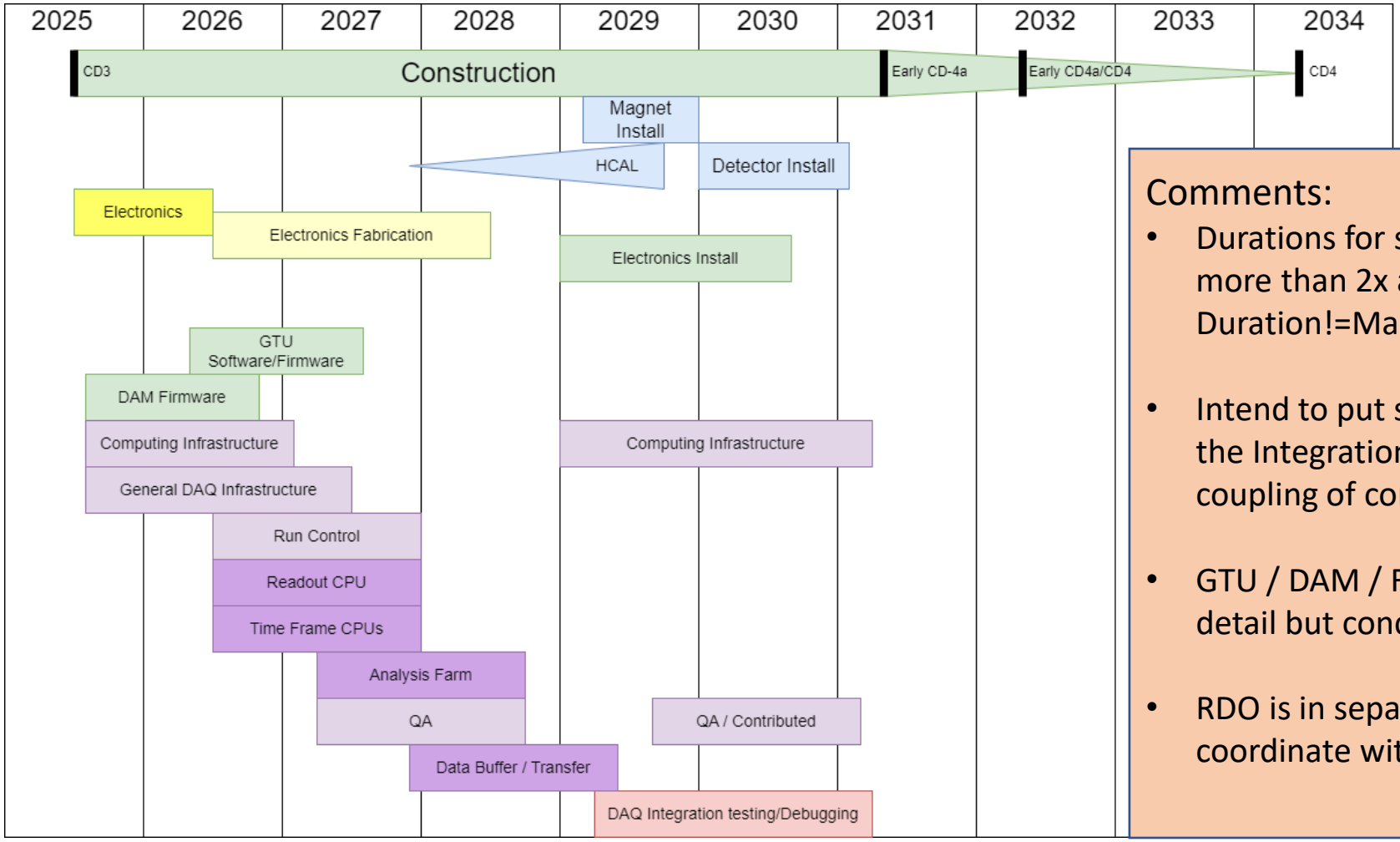
- The EVMS scheme defines a project in terms of work packages. There are monthly status updates towards completing the Work Packages, and when the work packages are done, the project is done! Guidance is →
    - Work packages should be ~1-2 years in duration (hours are separate from duration)
    - They should be composed of tasks lasting 30-60 days
    - Tasks need to have some sort of metric to apply, deliverables or at least judgement as to the fraction completed

Seems straightforward but there are number of issues

- Software not yet fully specified, and likely to change as needs become more clear
- DAQ systems components are tightly coupled, and must be defined and developed iteratively along with their connecting components
- There is lots of existing code (CODA, sPHENIX DAQ, STAR DAQ, etc...) but nothing that satisfies fully the needs → tasks may involve significant evaluation and adaptation and its not clear yet what tasks are development and which are adaptation.

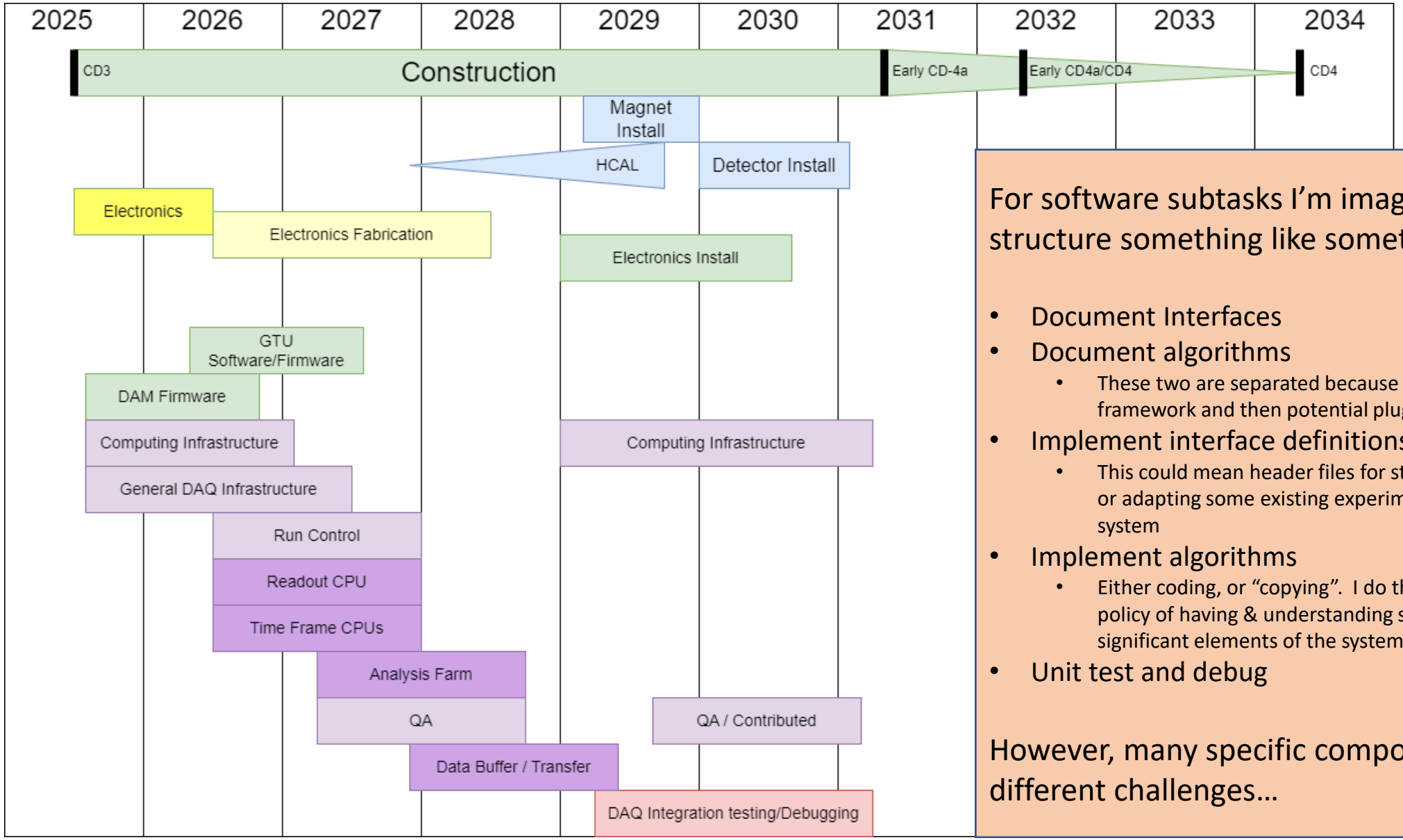
Strategy

The goal is to get something complete that is flexible enough to handle the types of likely activities we will need while being detailed enough to make sure that checking off the WBS work package milestones results in a full, operating DAQ system



Comments:

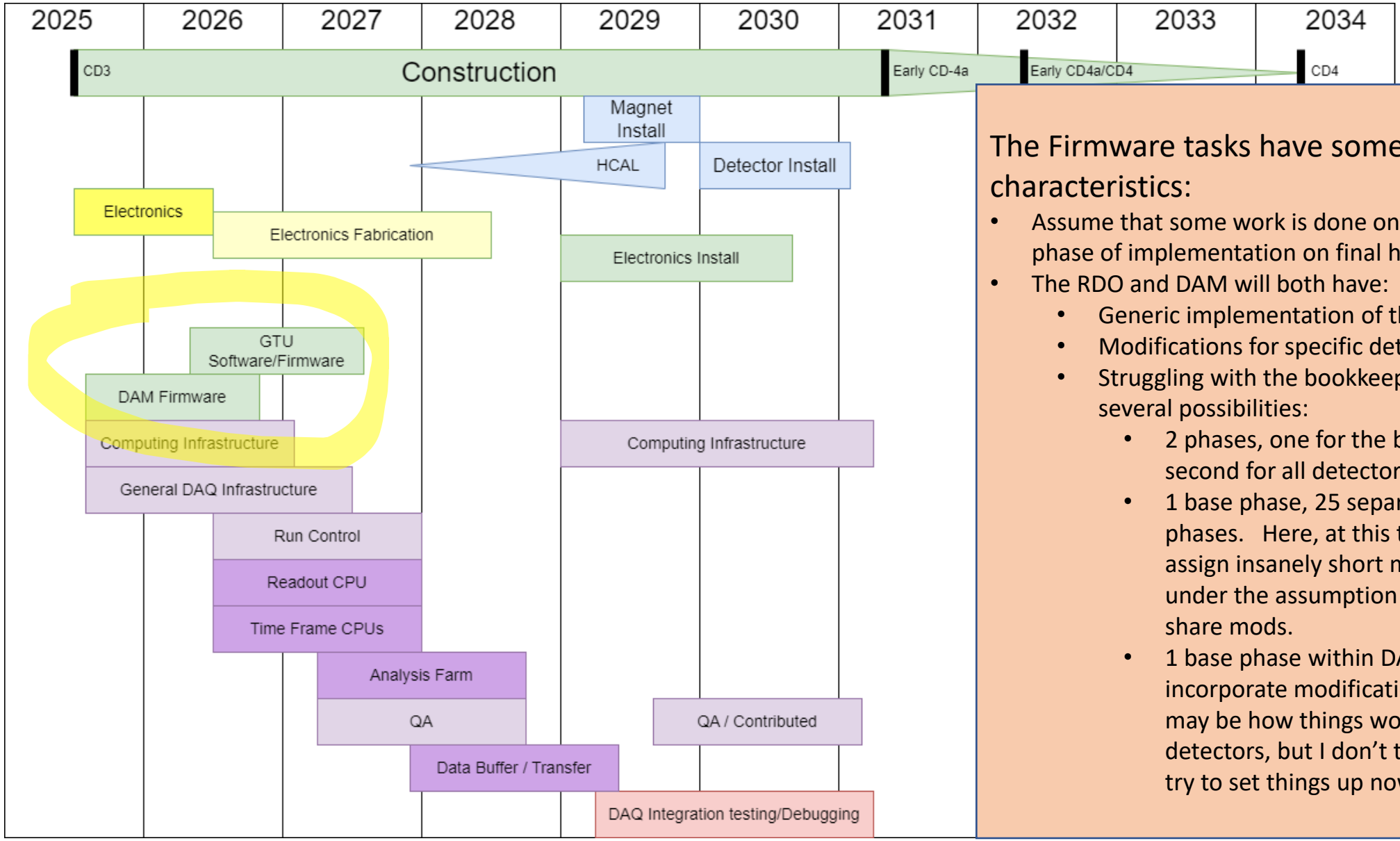
- Durations for software/firmware add up to a bit more than 2x actual allocated FTE. Duration!=Manpower
- Intend to put significant manpower ~25% into the Integration testing/Debugging phase due to coupling of components
- GTU / DAM / Fiber Runs also included in more detail but concentrate on software right now…
- RDO is in separate electronics CAM, so need to coordinate with Fernando



For software subtasks I'm imagining a general structure something like something like:

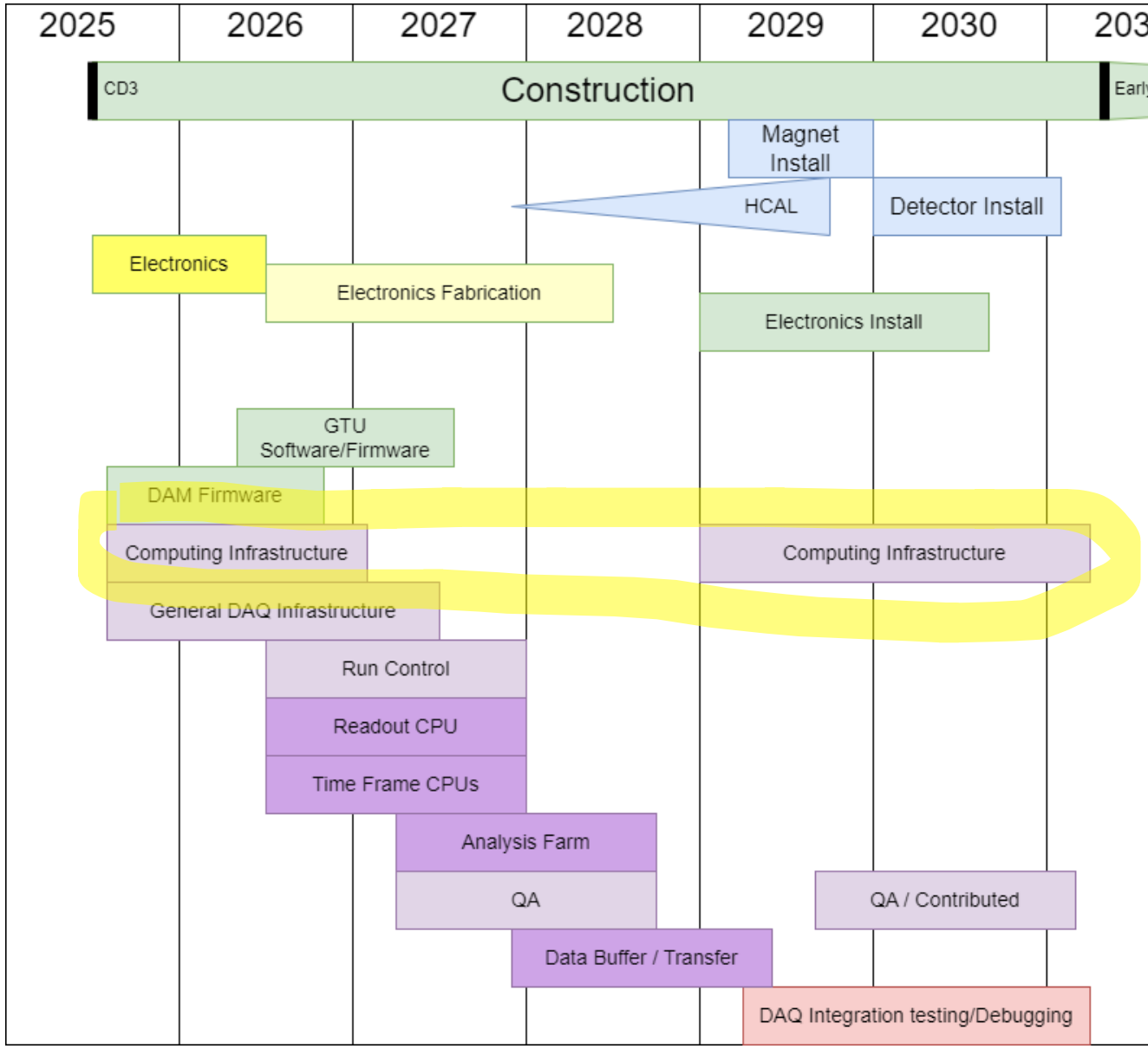
- Document Interfaces
- Document algorithms
  - These two are separated because many items have a framework and then potential plug-ins for specific work
- Implement interface definitions
  - This could mean header files for straight development work or adapting some existing experiments code to the ePIC system
- Implement algorithms
  - Either coding, or "copying". I do think we need to adopt the policy of having & understanding source code for all significant elements of the system
- Unit test and debug

However, many specific components have different challenges…



The Firmware tasks have some special characteristics:

- Assume that some work is done on test boards and needs a phase of implementation on final hardware
- The RDO and DAM will both have:
  - Generic implementation of the structure
  - Modifications for specific detectors (typically quick?)
  - Struggling with the bookkeeping aspect among several possibilities:
    - 2 phases, one for the base code, and the second for all detector specific modifications.
    - 1 base phase, 25 separate modification phases. Here, at this time, we'd have to assign insanely short modification phases, under the assumption many detectors would share mods.
    - 1 base phase within DAQ, ask detectors to incorporate modifications themselves. This may be how things work in practice for many detectors, but I don't think it's a helpful way to try to set things up now.

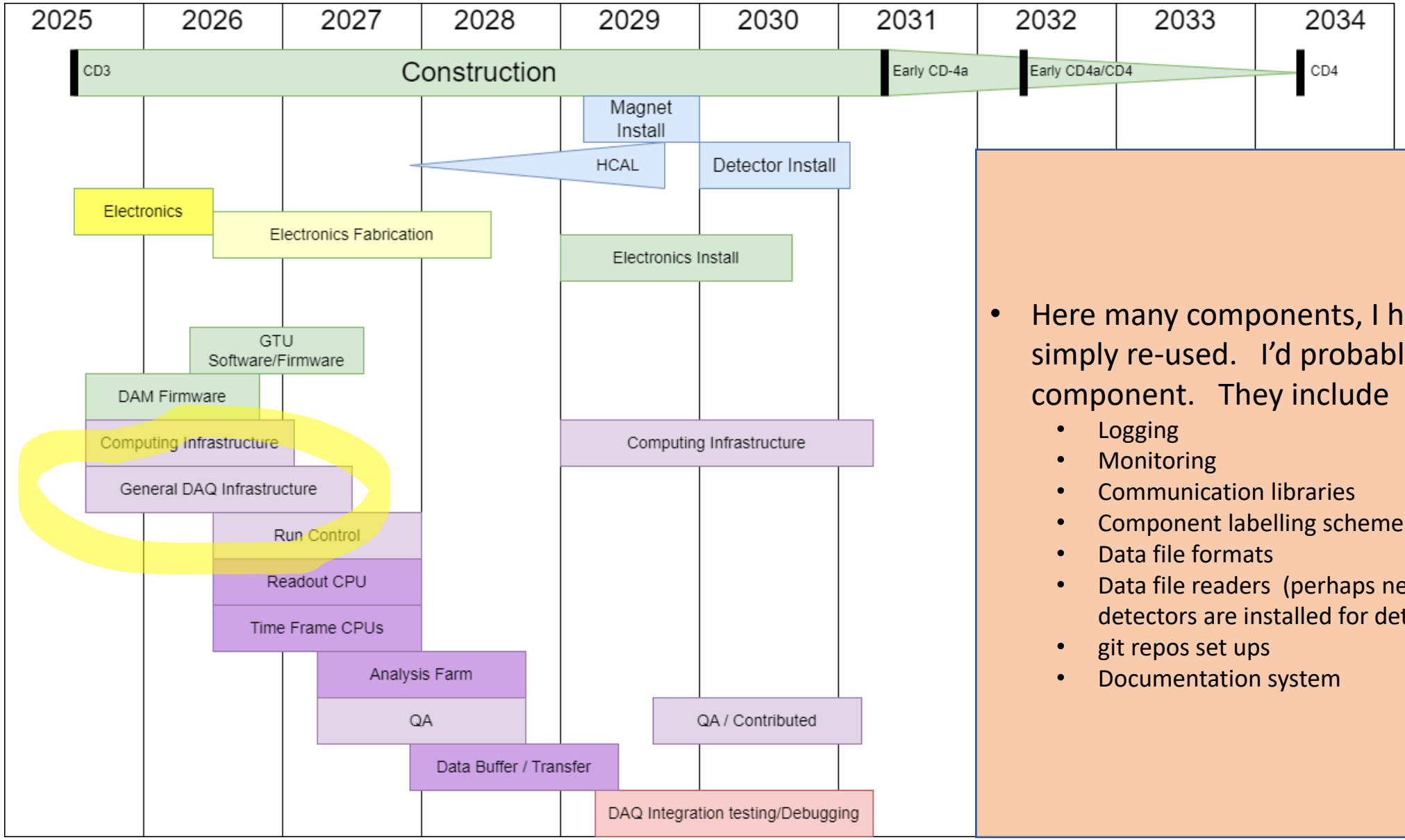


Computing Infrastructure is in two phases:

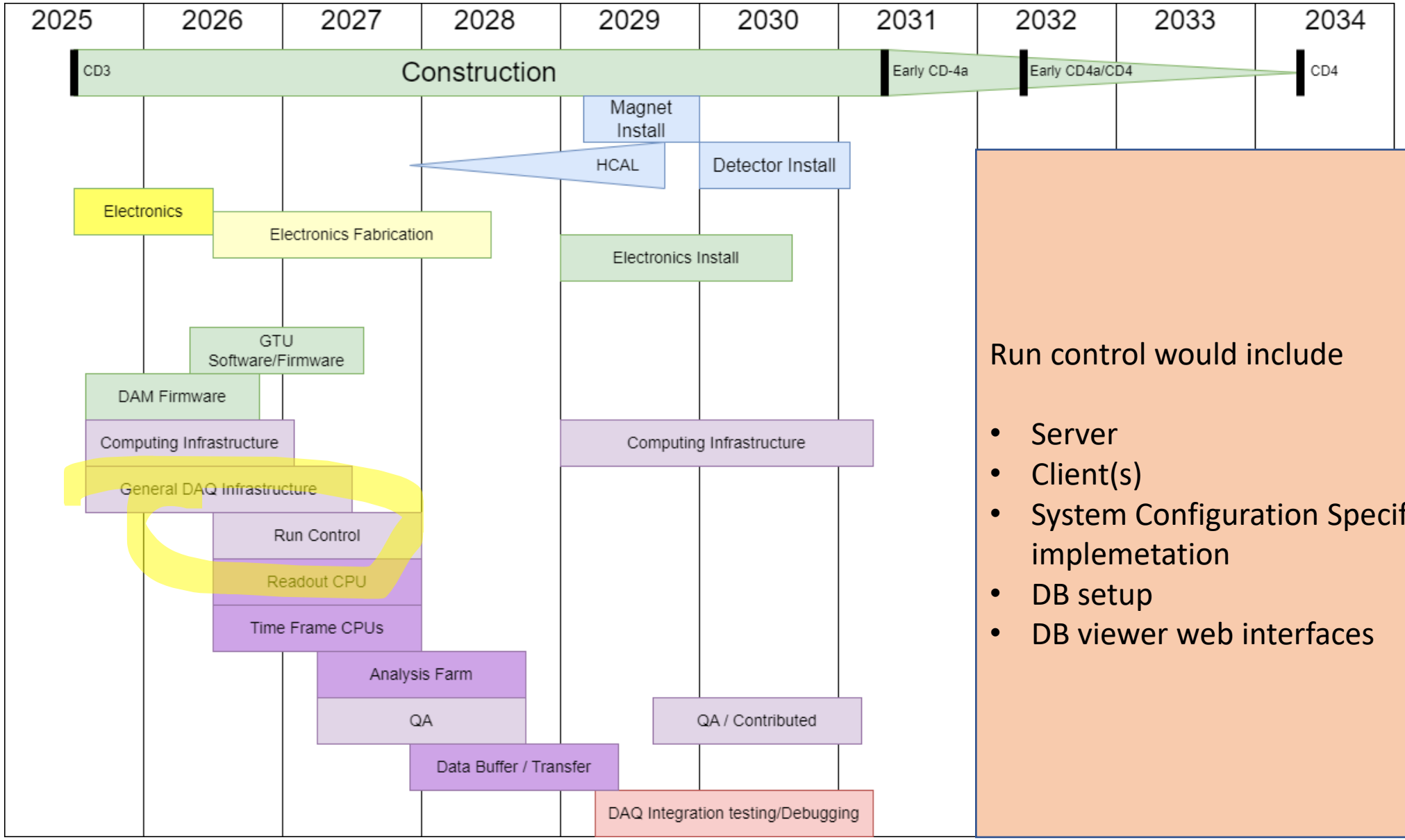
- Phase 1: Network, Main DAQ servers, Development servers (some potentially reused from STAR/Phenix)
- Phase 2: Final full computing farm

The phases of this item are also more unique. The tasks include

- Procurement
- Installation
- System admin / network admin
- OS installation / Management software (network boot) configuration
- Little if any coding development

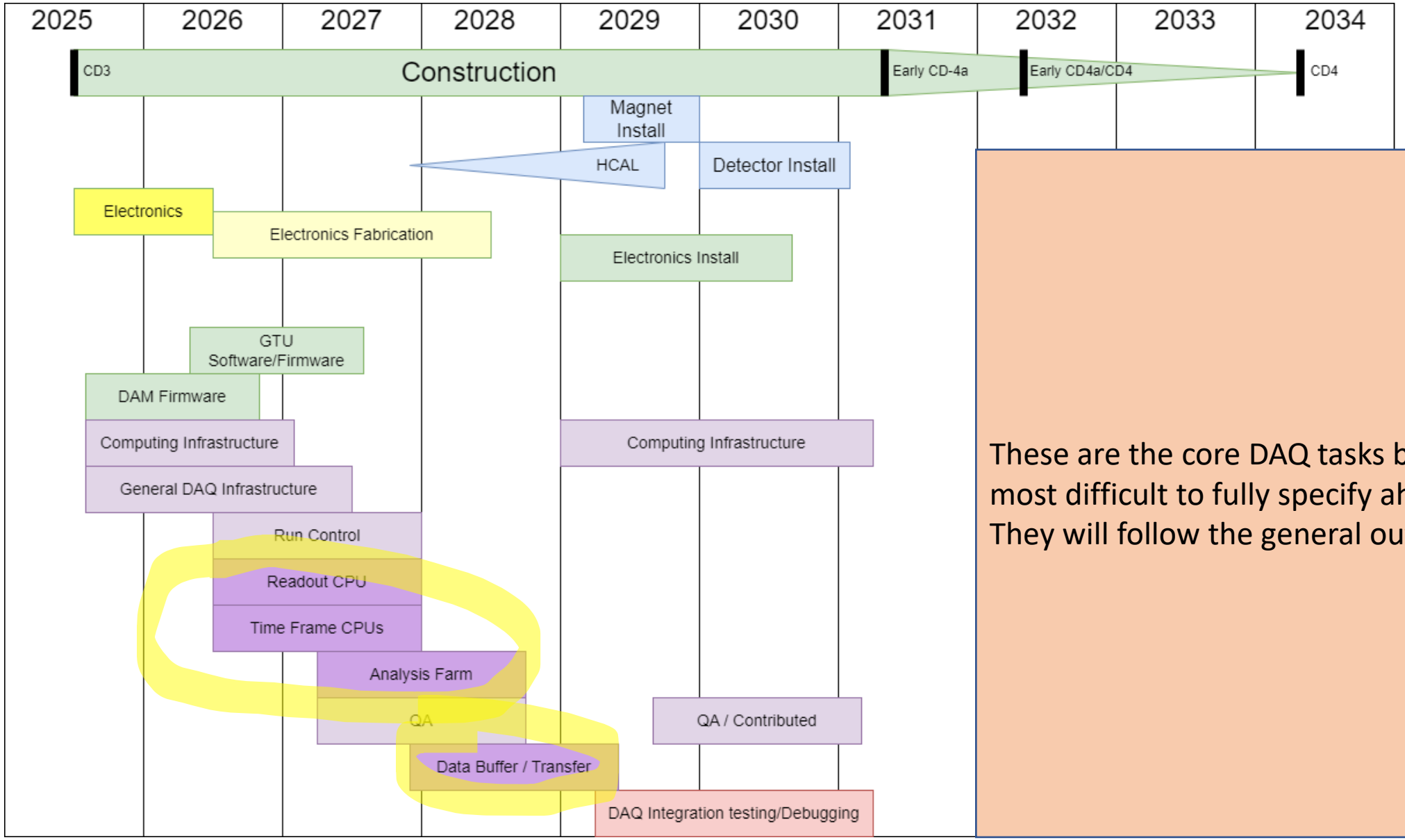


- Here many components, I hope many are simply re-used. I'd probably put one task per component. They include
  - Logging
  - Monitoring
  - Communication libraries
  - Component labelling scheme
  - Data file formats
  - Data file readers (perhaps need a phase2 when detectors are installed for detector bank readers)
  - git repos set ups
  - Documentation system



Run control would include

- Server
- Client(s)
- System Configuration Specification and implemetation
- DB setup
- DB viewer web interfaces



These are the core DAQ tasks but they are the most difficult to fully specify ahead of time. They will follow the general outline

I’ve outlined the set of work packages and subtasks

What I’m trying to do is a sort of though experiment in which the years 2025-2031 are laid out in 1 month chunks. I’ve tried to describe the bullet items that are going to end up on the calendar for each of those months.

The question is whether these bullets seem doable, and whether they lead to a functioning DAQ if done?

Questions, Queries, Suggestions?