



ALICE

Adapting to Change: A look at the evolution of ALICE's Quality Control framework

21.10.2024

Piotr Konopka, Barthélémy von Haller, Michal Tichak
on behalf of the ALICE Collaboration

Simplified software development process

Requirements

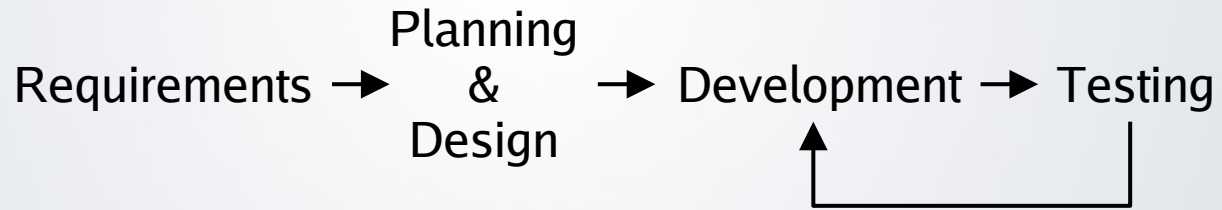
Simplified software development process

Requirements → Planning
&
Design

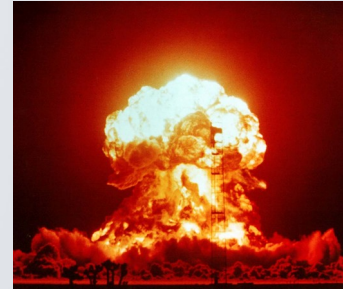
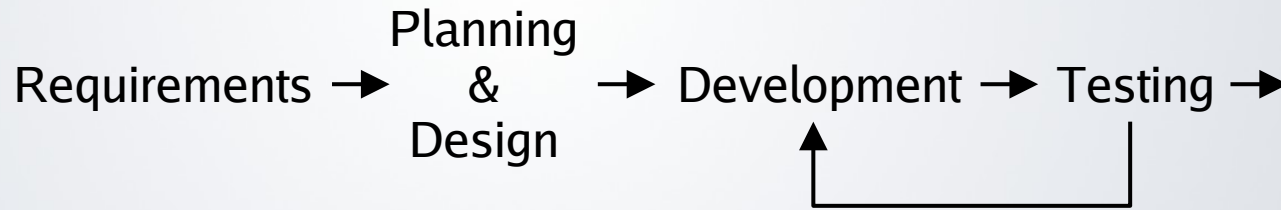
Simplified software development process



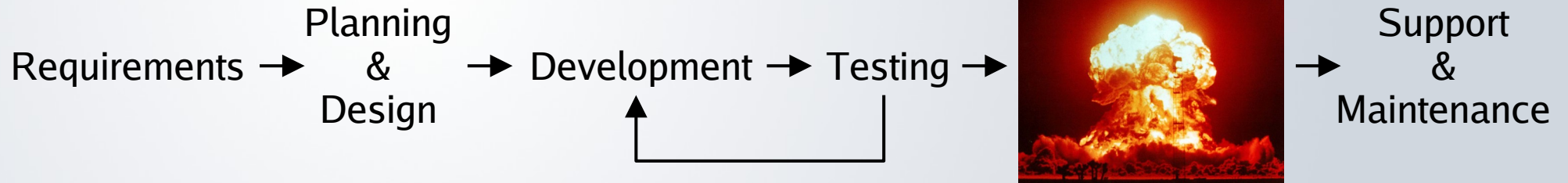
Simplified software development process



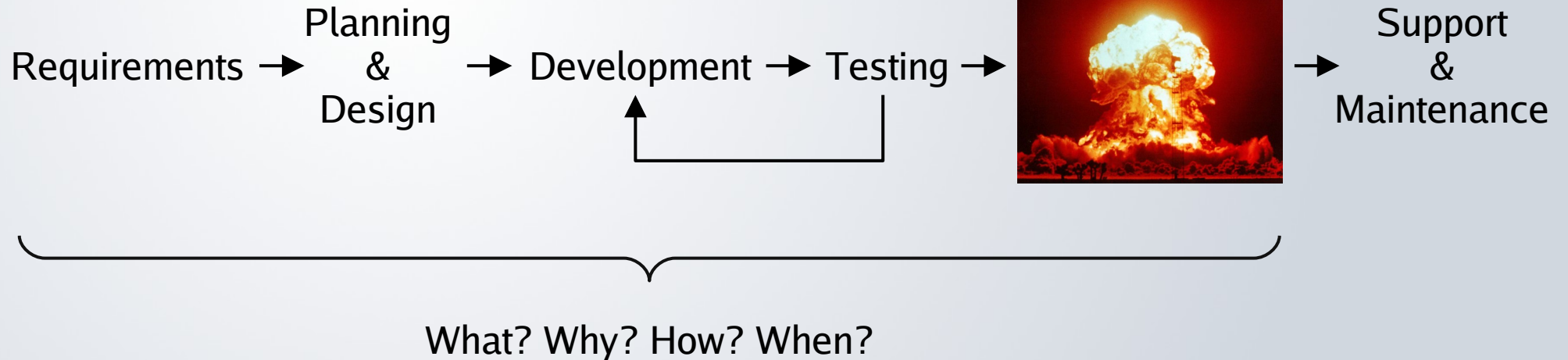
Simplified software development process



Simplified software development process



Simplified software development process



The ALICE data Quality Control framework



Upgrade of the Online - Offline computing system

Technical Design Report




Data Quality Control evolution in ALICE

LHC Run 1&2:
(until 2018)

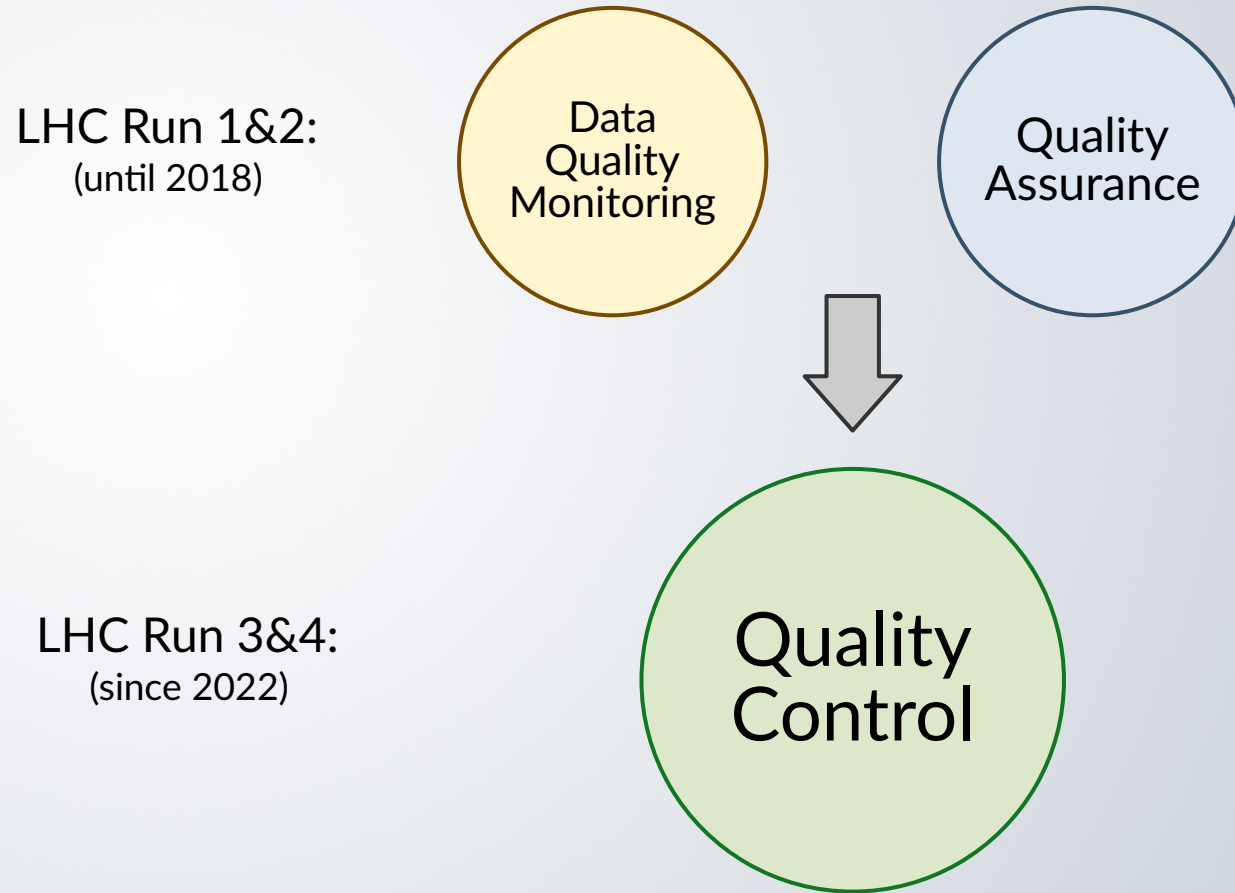


Data
Quality
Monitoring

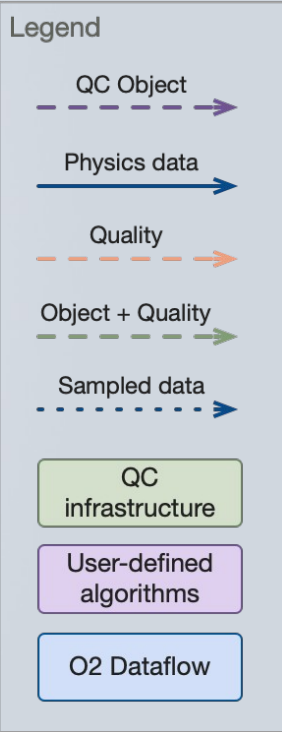
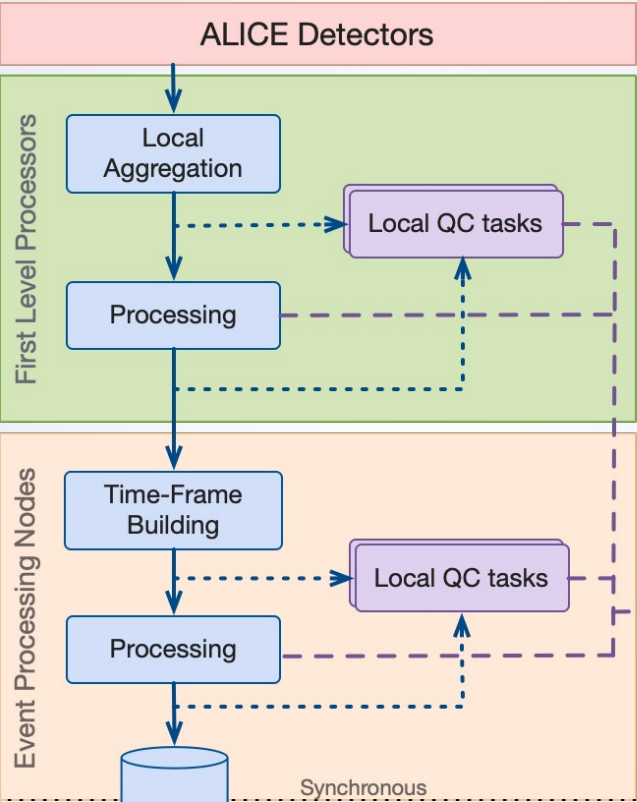


Quality
Assurance

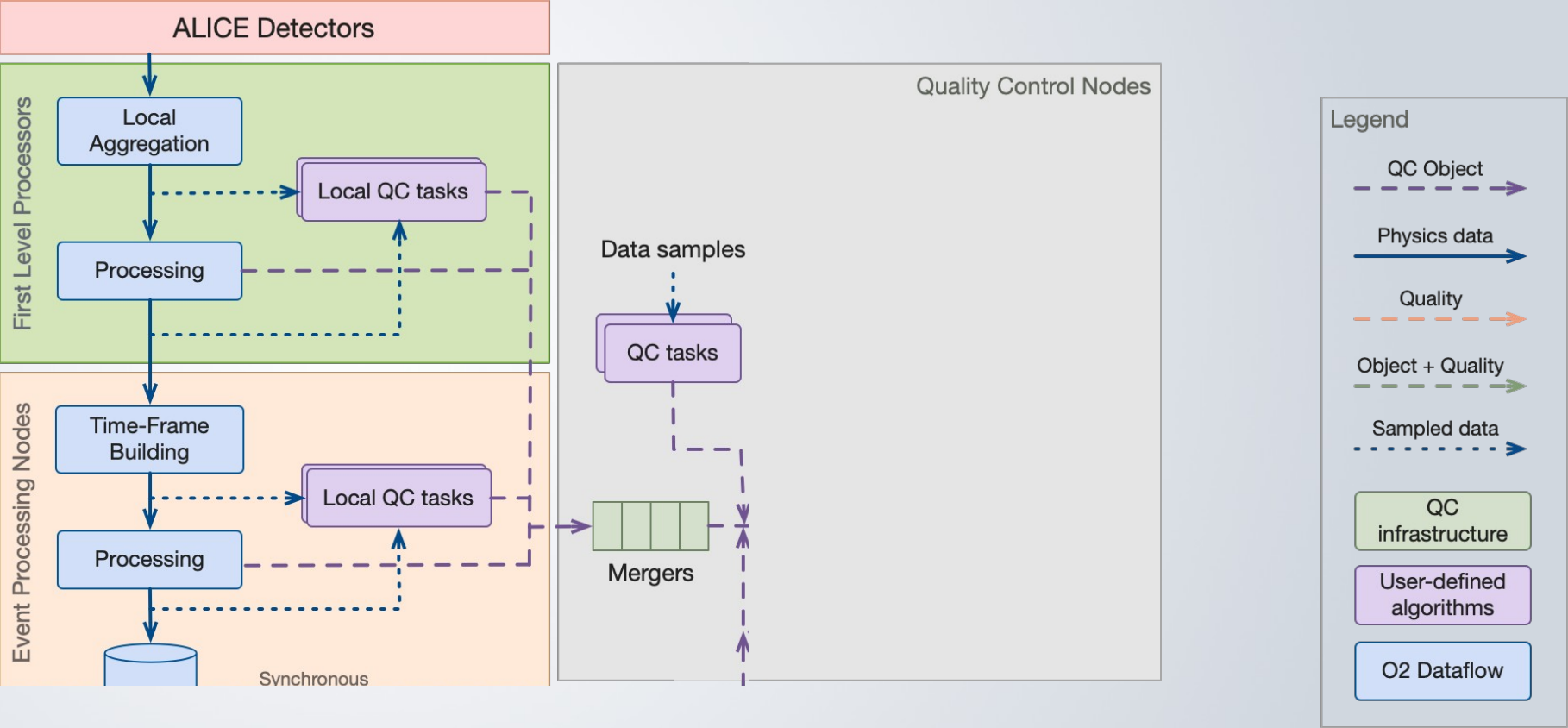
Data Quality Control evolution in ALICE



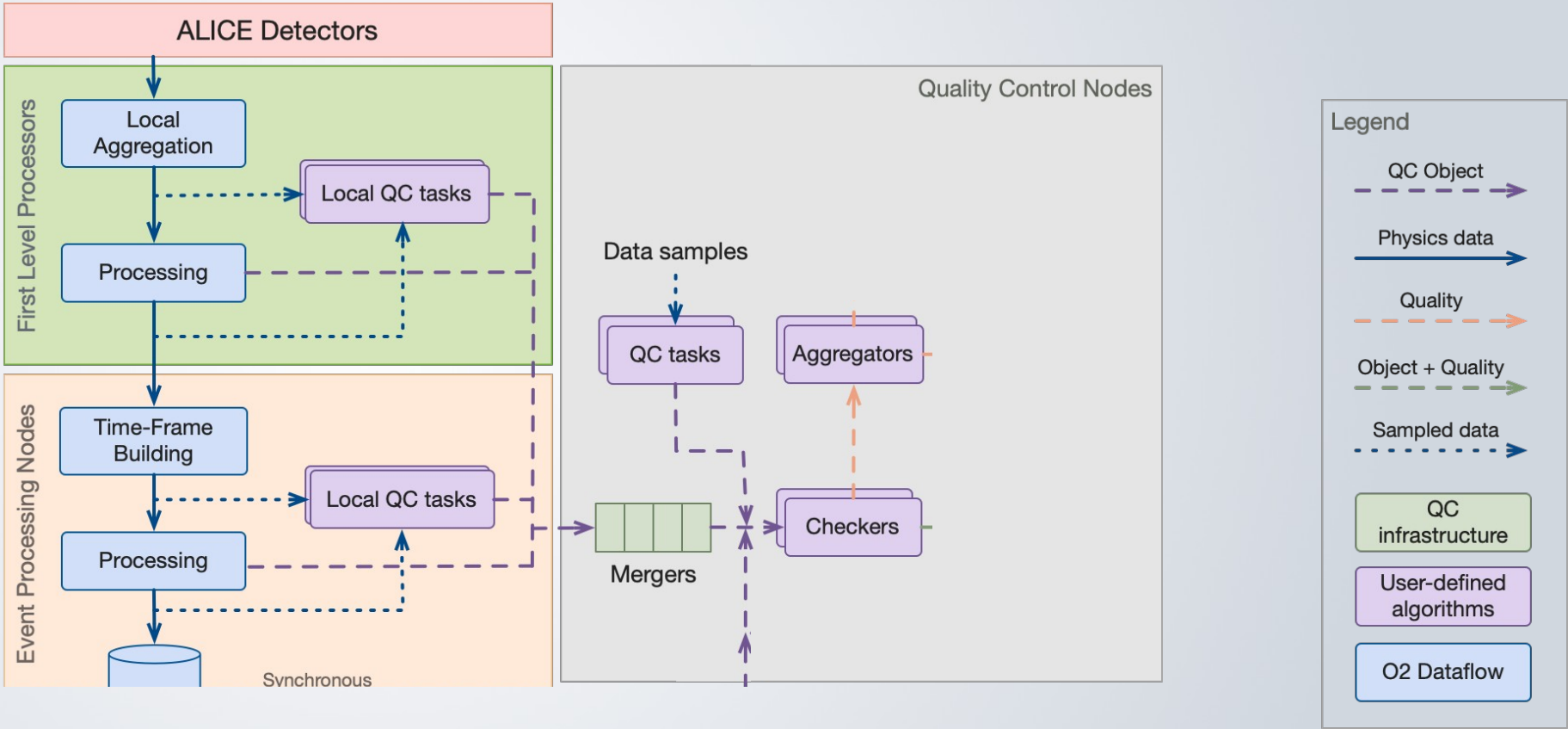
Quality Control framework



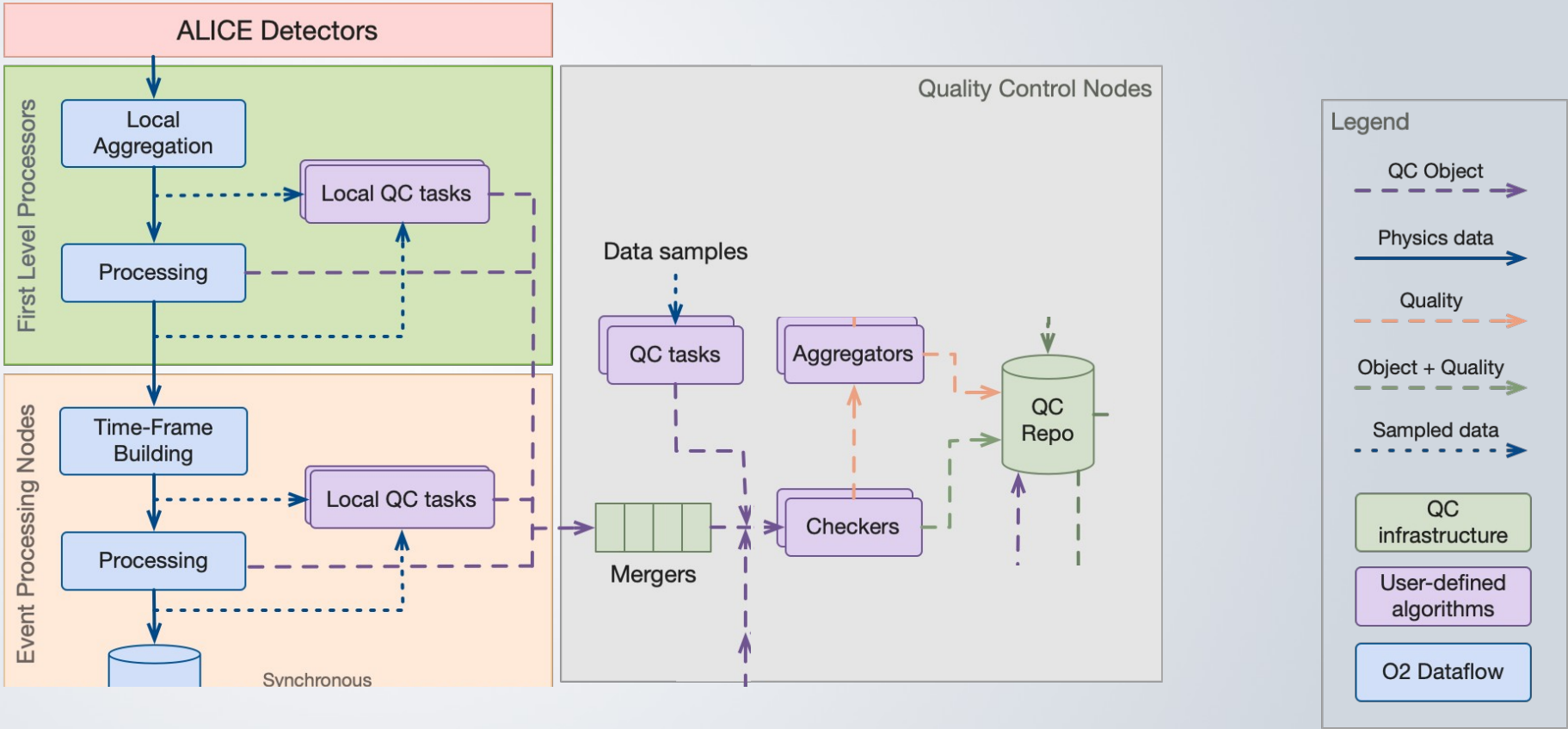
Quality Control framework



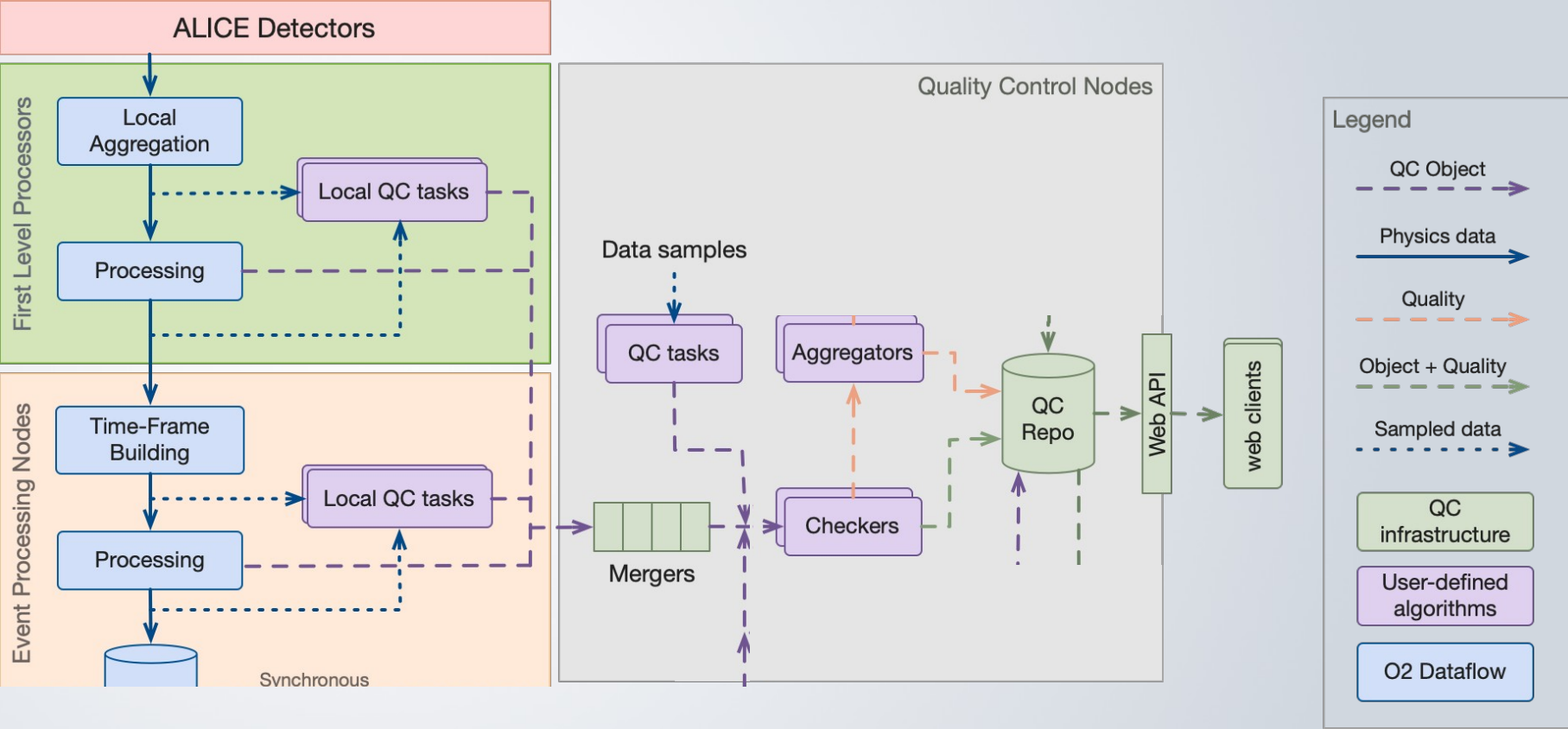
Quality Control framework



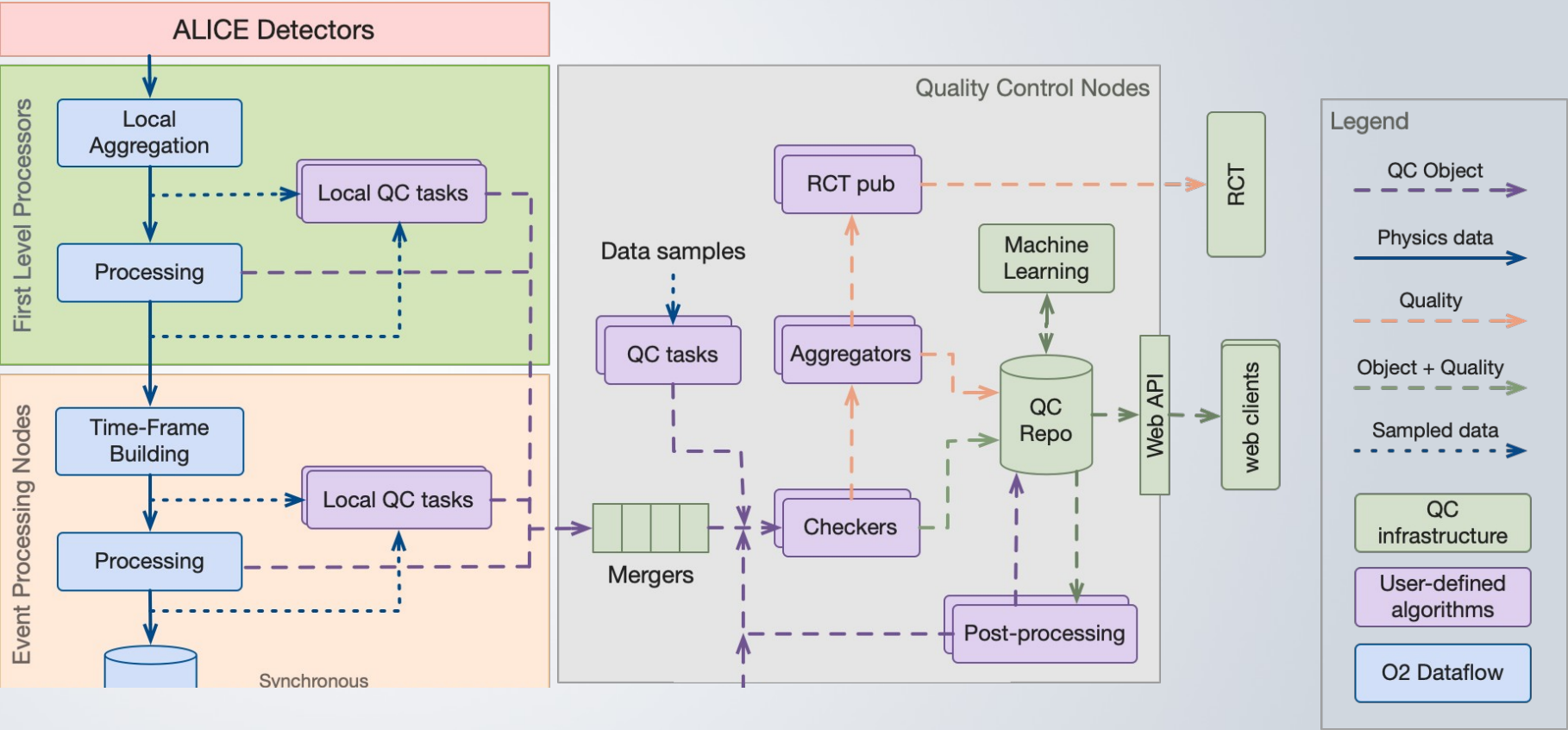
Quality Control framework



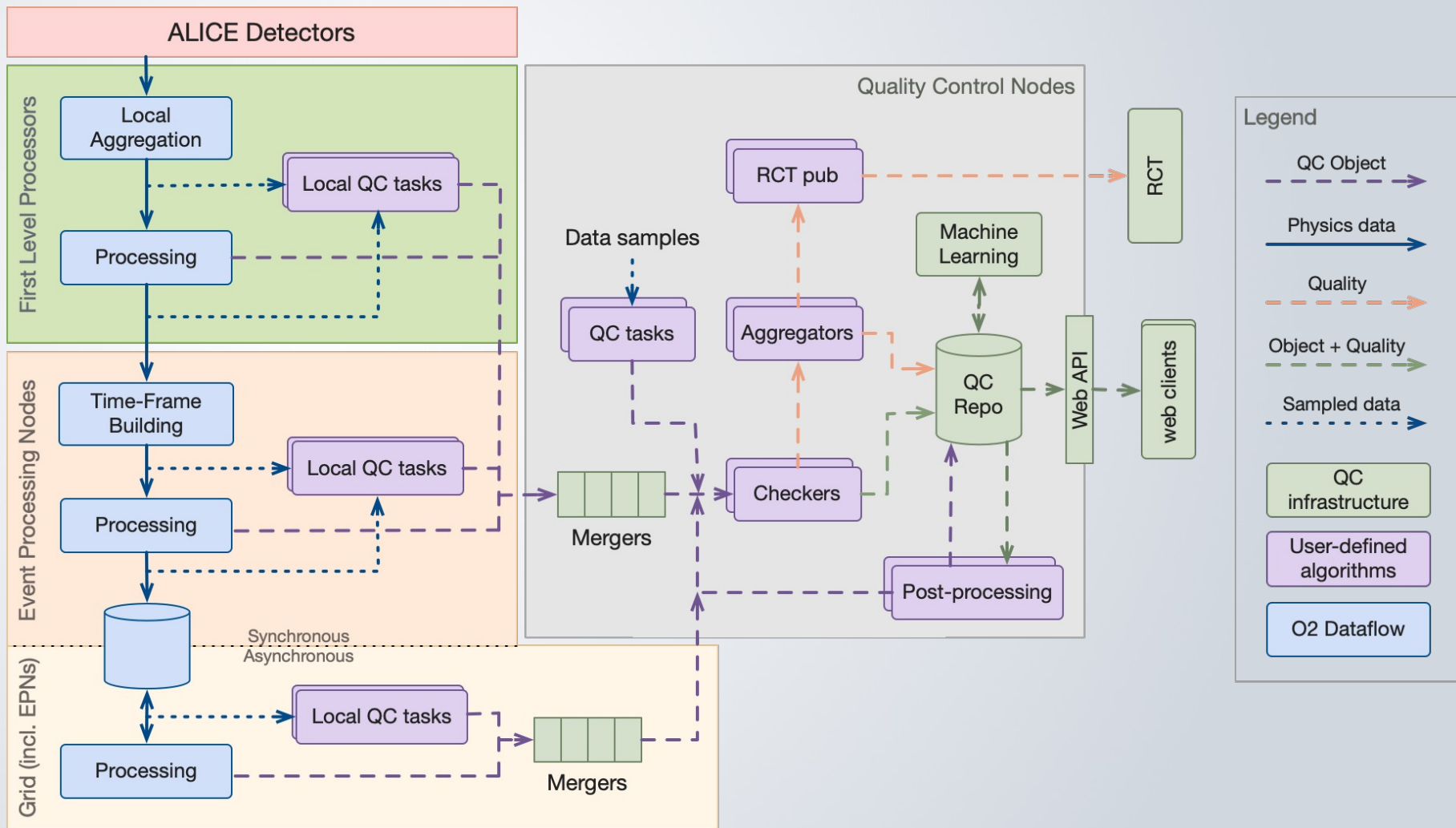
Quality Control framework



Quality Control framework

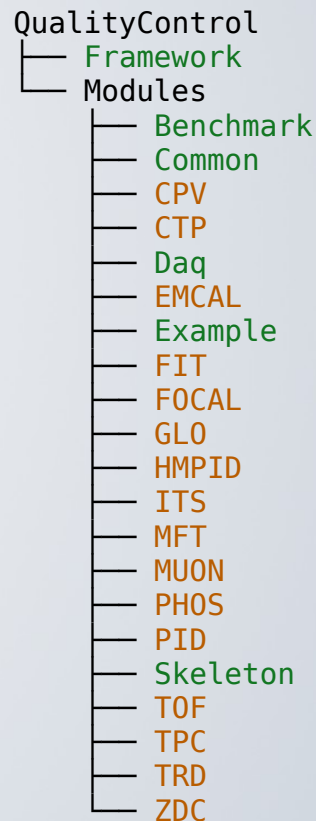


Quality Control framework



Responsibilities

- **Developer** team responsible for the framework and common modules
- Detector modules written by **detector teams**

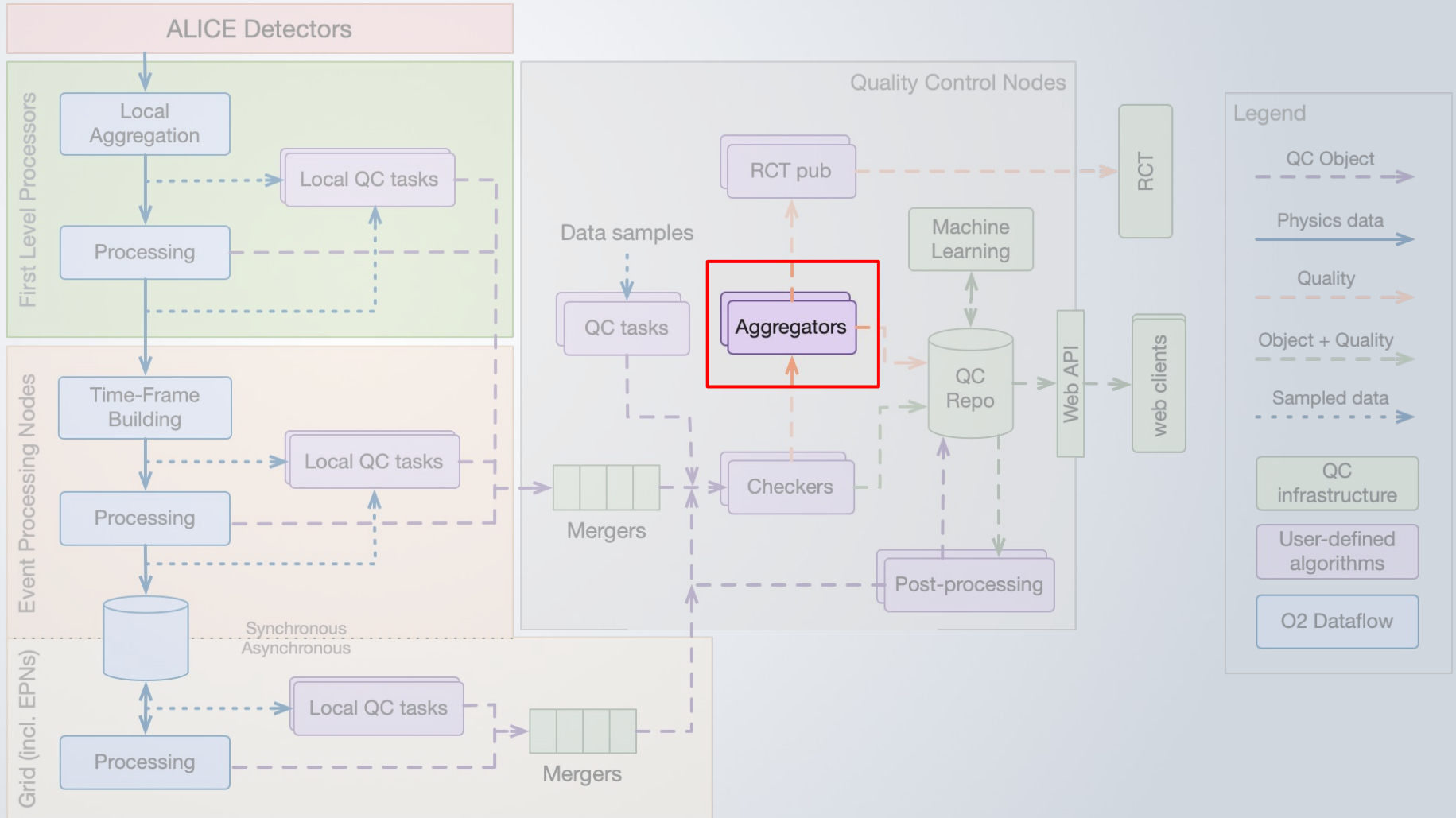


From the beginning

Gathering requirements

- Project kickoff in 2013
- Regular meetings since 2014 to discuss features and performance needs
- Future users are (obviously) contained in their domains
- Users knowing how to program often propose implementations
- The most valuable feedback came from experienced users of similar systems before
- Most feature requests come once users start using a piece of software

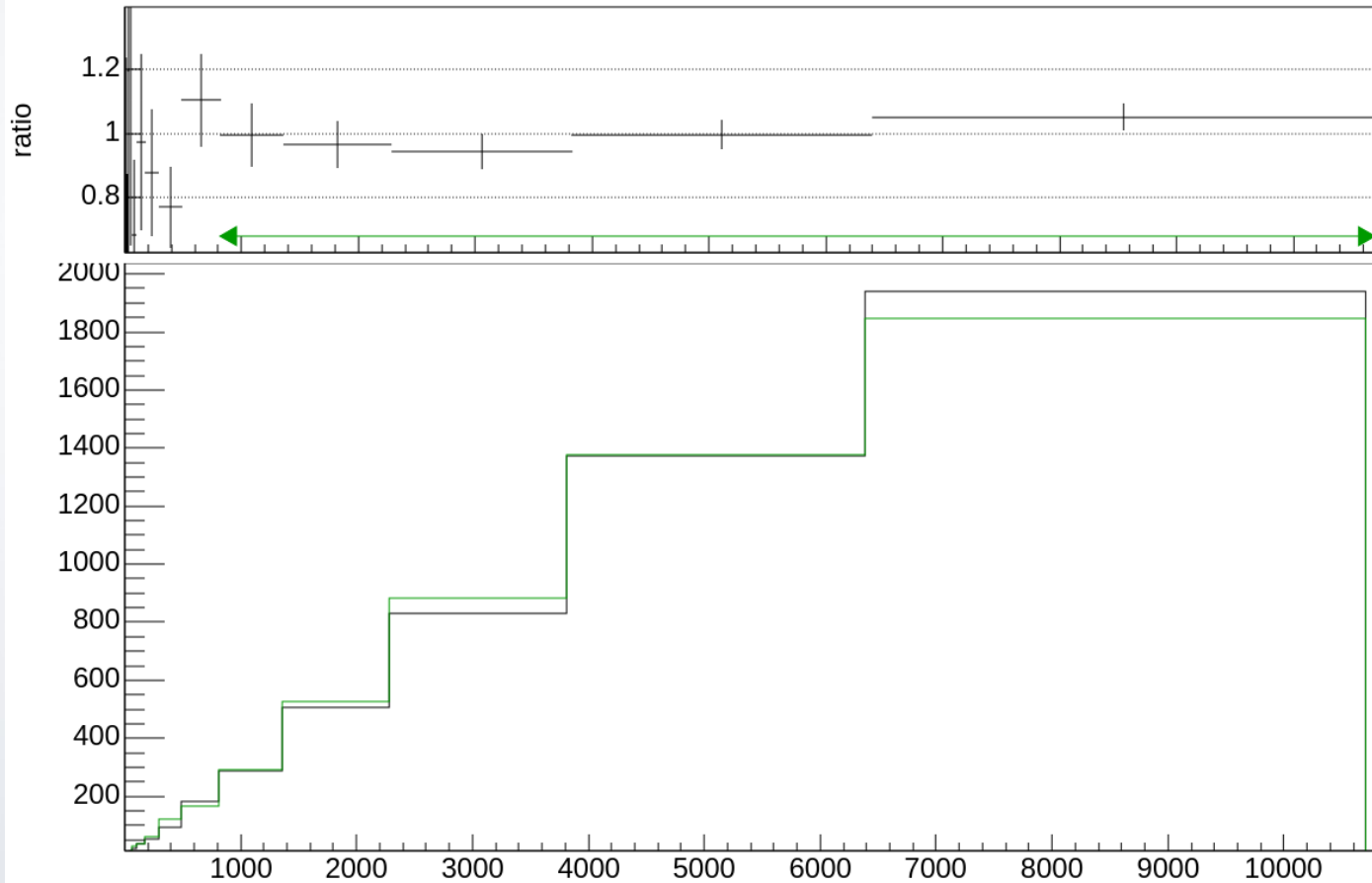
Late requirements - quality aggregation



Late requirements - reference data

Good

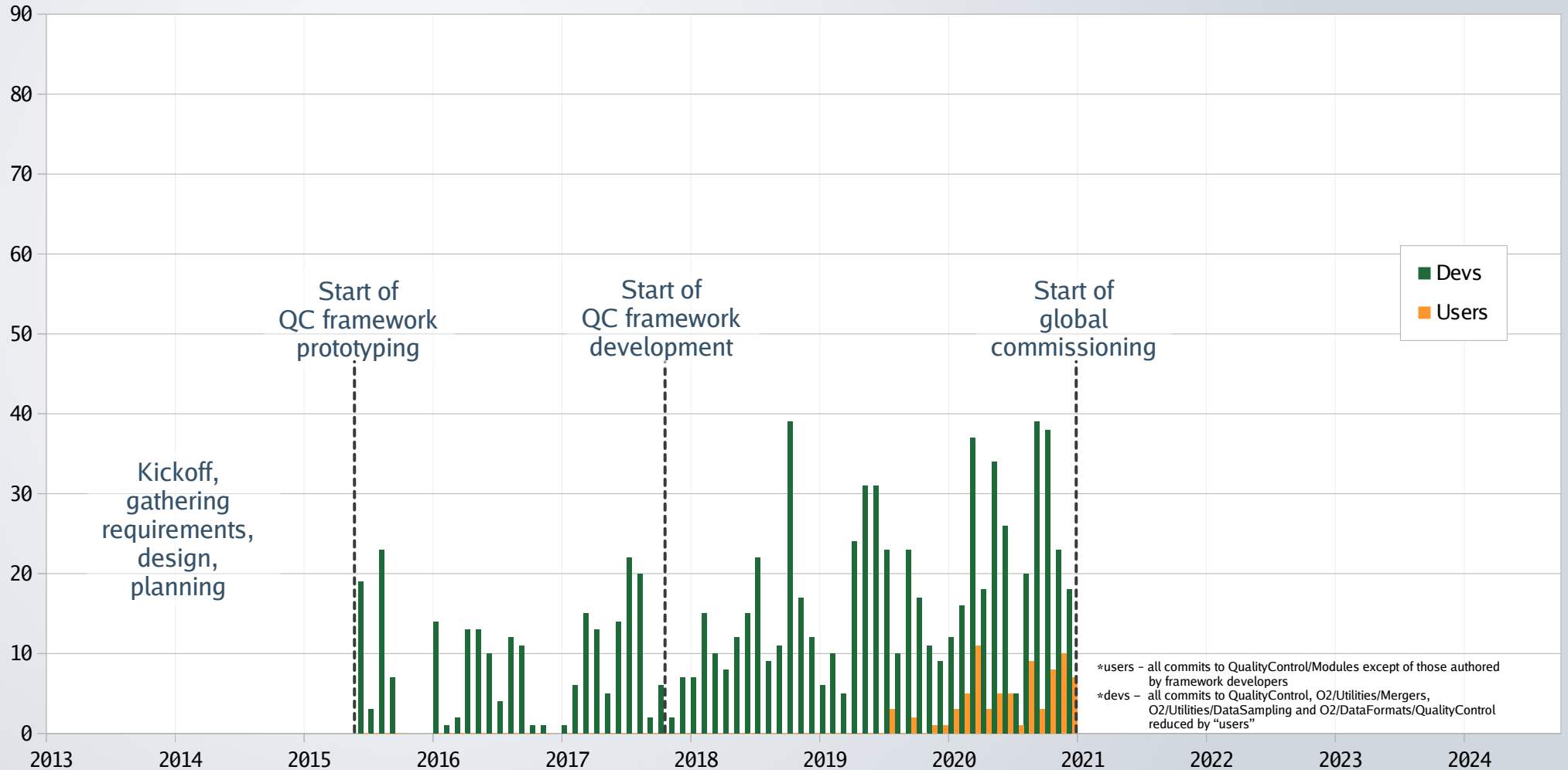
example



Performance requirements – number of objects

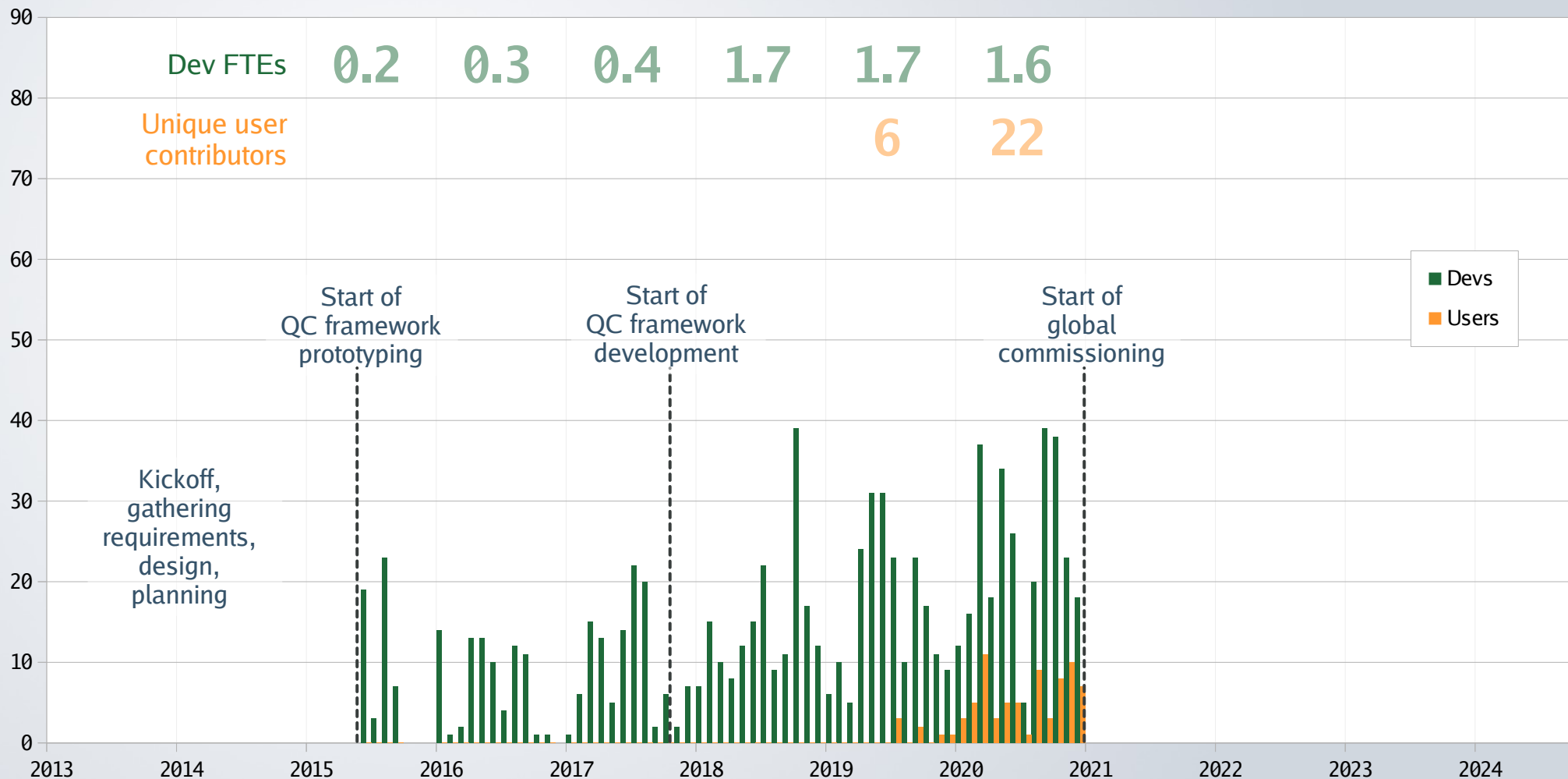
	2014 (estimates)	2020 (estimates)	2022 (real)	2024 (real)
DET A	4	23055	42	149
DET B	9	3	3315	906
DET C	70	3690	289	643
...
Total	5000	52000	12644	28836

Commit frequency in QC until the start of global commissioning



*users - all commits to QualityControl/Modules except of those authored by framework developers
*devs - all commits to QualityControl, O2/Utilities/Mergers, O2/Utilities/DataSampling and O2/DataFormats/QualityControl reduced by "users"

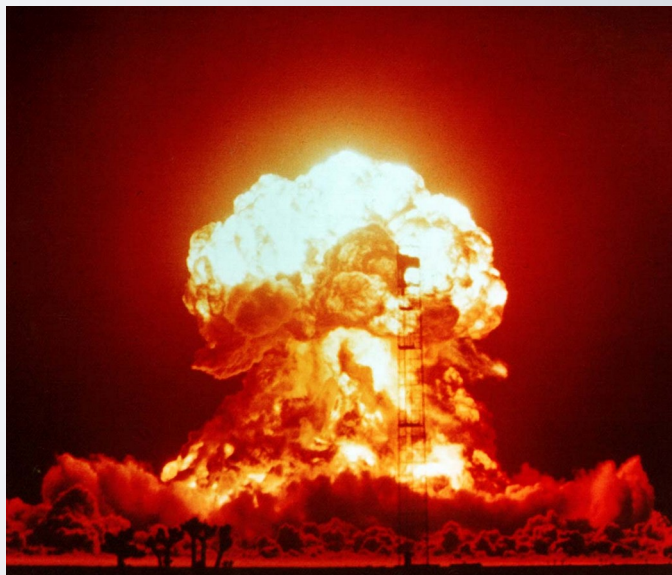
Commit frequency in QC until the start of global commissioning



Framework adoption

- Early adopters help to smoothen the workload peak during commissioning
 - Difficult: no data to test, few examples, unfamiliarity, other priorities
- Advertising new features:
 - Release notes alone are not sufficient
 - Important features presented in meetings
 - Regular reviews of detector QC status
- Form a community, so detector teams can share knowledge between each other
- Long-time users are able to contribute to the framework

2021 - start of



Milestone Weeks during Global Commissioning

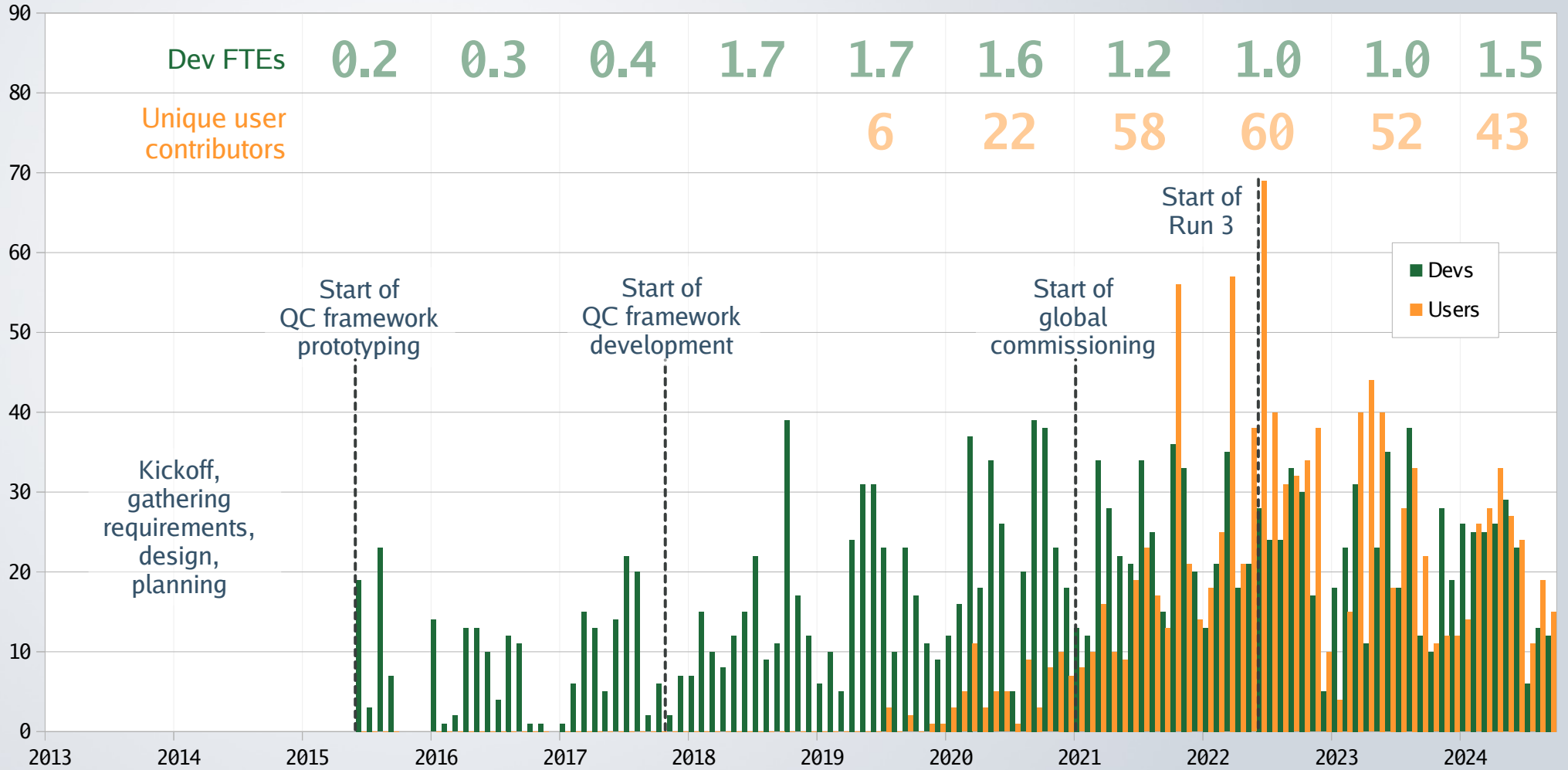
- One week each month dedicated for integration and commissioning of ALICE
- Helpful, because:
 - we had concrete, results-oriented goals to work towards
 - we could progress
- Disrupting, because:
 - A lot of time used for testing and reacting, reducing available development time
 - We had to develop quickly, thus accumulate technological debt

Bootstrapping a data acquisition ecosystem

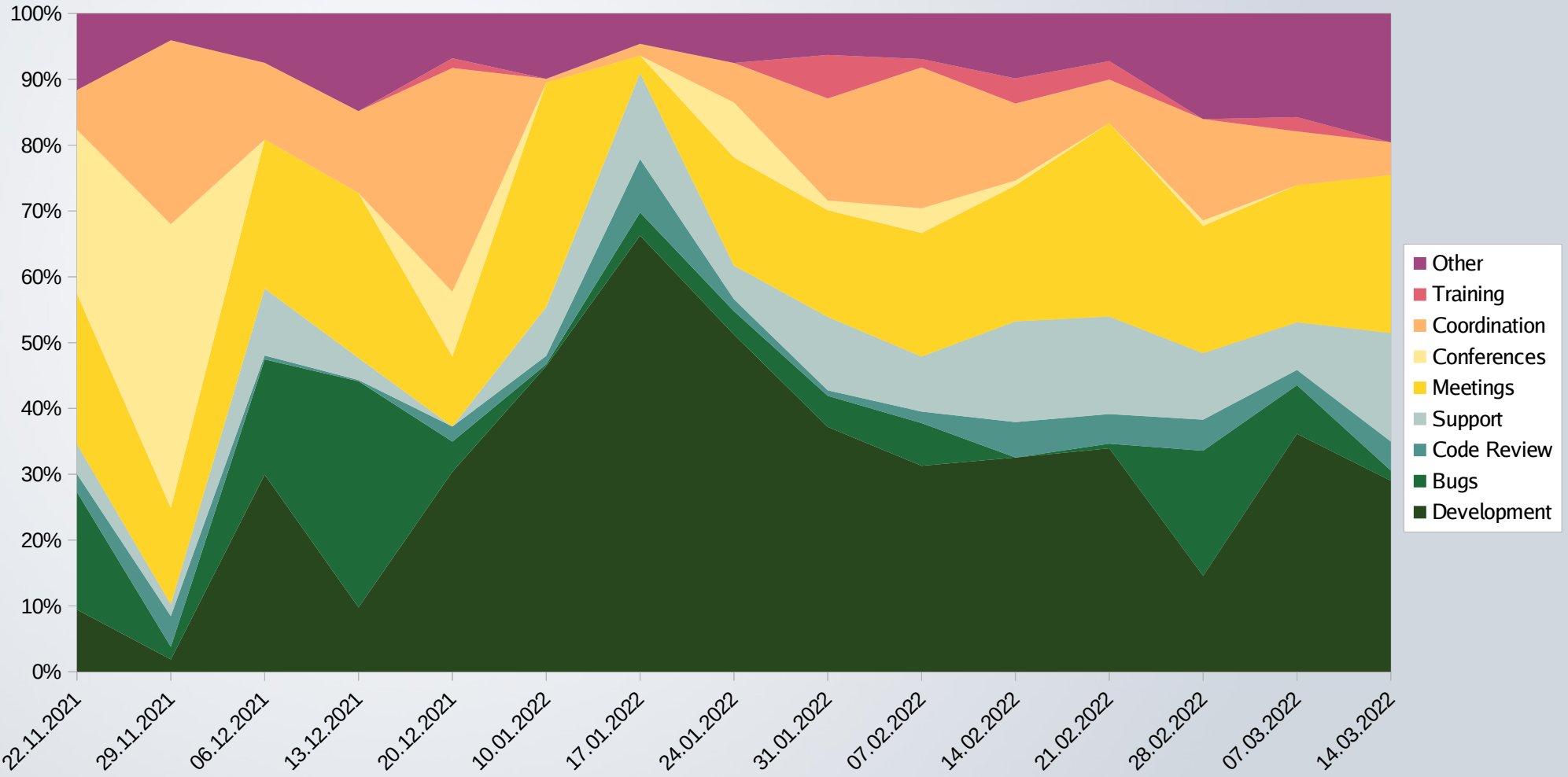


<https://www.cbc.ca/news/canada/prince-edward-island/pei-roof-built-on-ground-1.4556019>

Commit frequency in QC



Work time shares per week (Nov 2021 – Mar 2022)

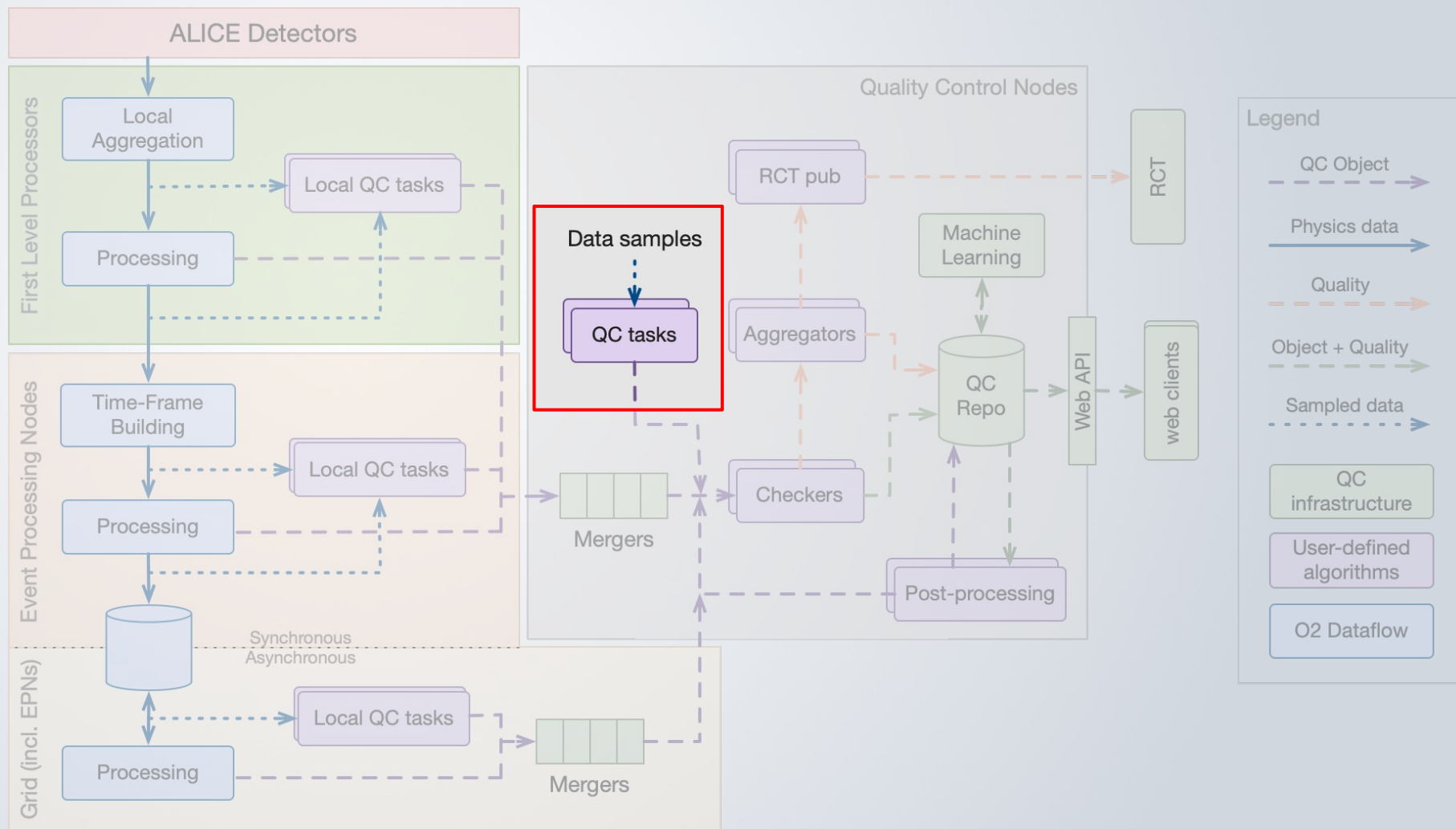


Once the dust has settled

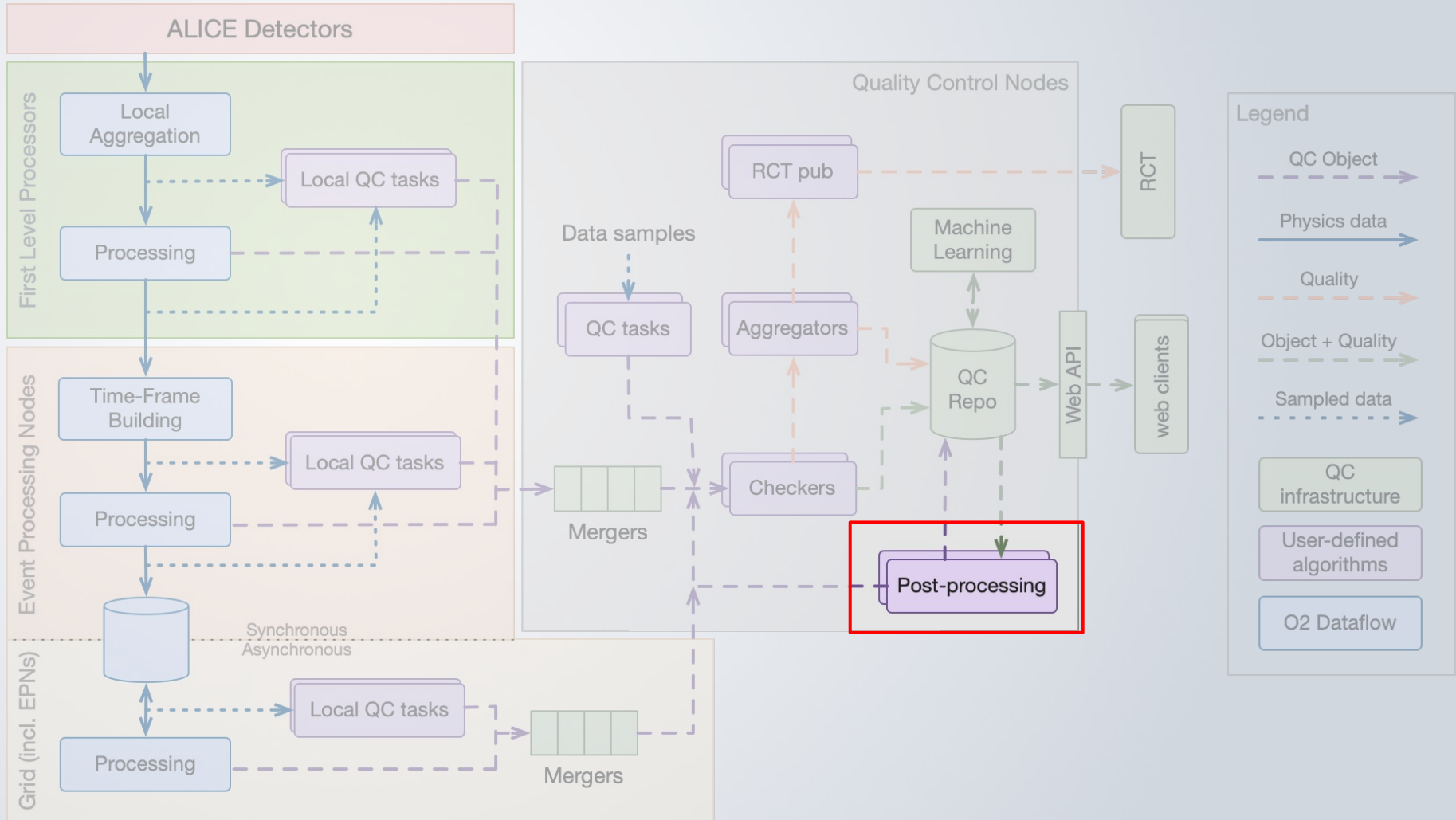
Main achievements

- Merged user teams which previously worked separately
- All of the major requirements were fulfilled
 - One data quality system for online and offline use
 - QC can be plugged at any processing stage, to any data
 - Automatic Checks
 - Results accessible worldwide
 - Scalable, high-performance framework

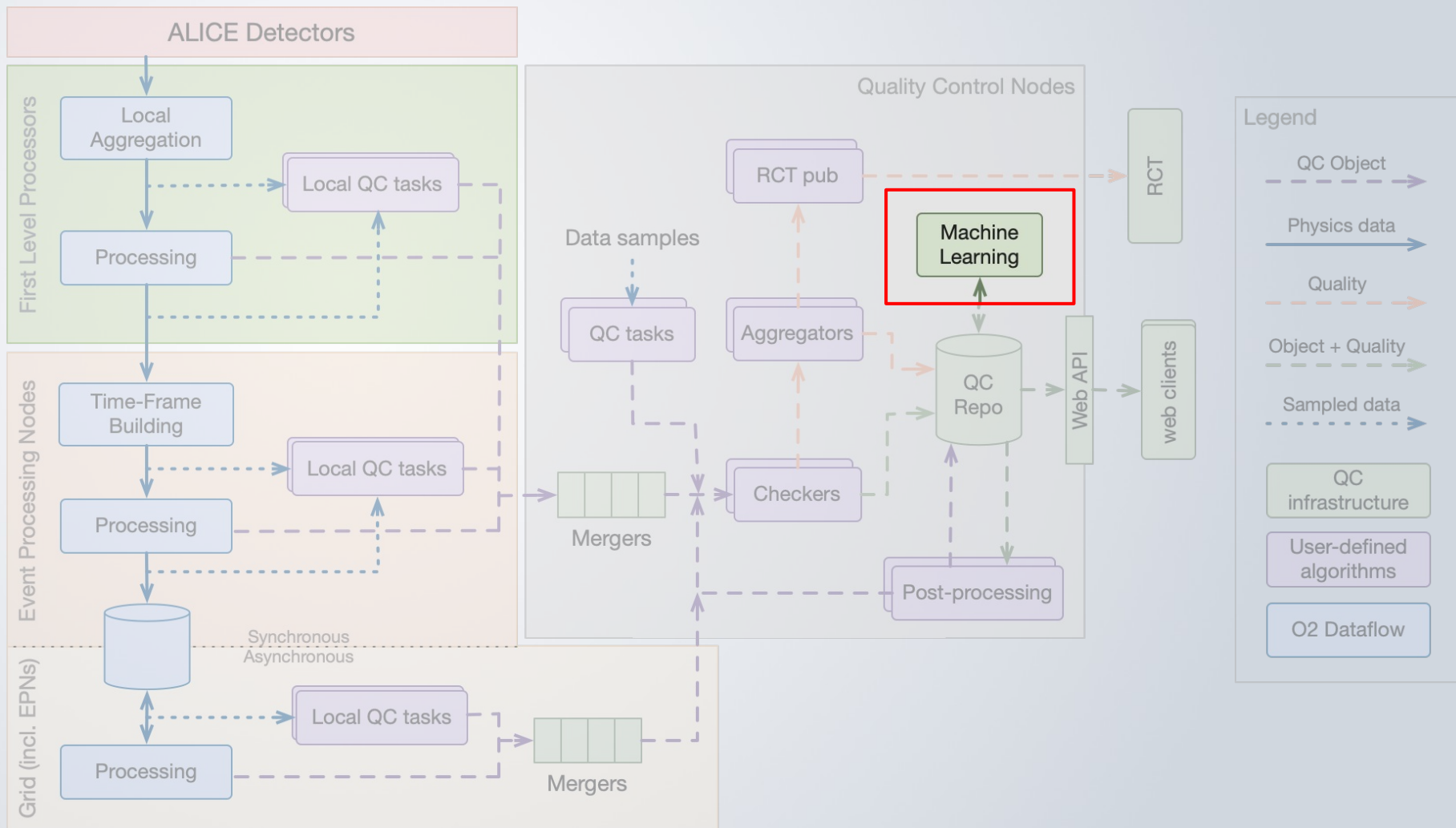
Features not used - remote QC tasks



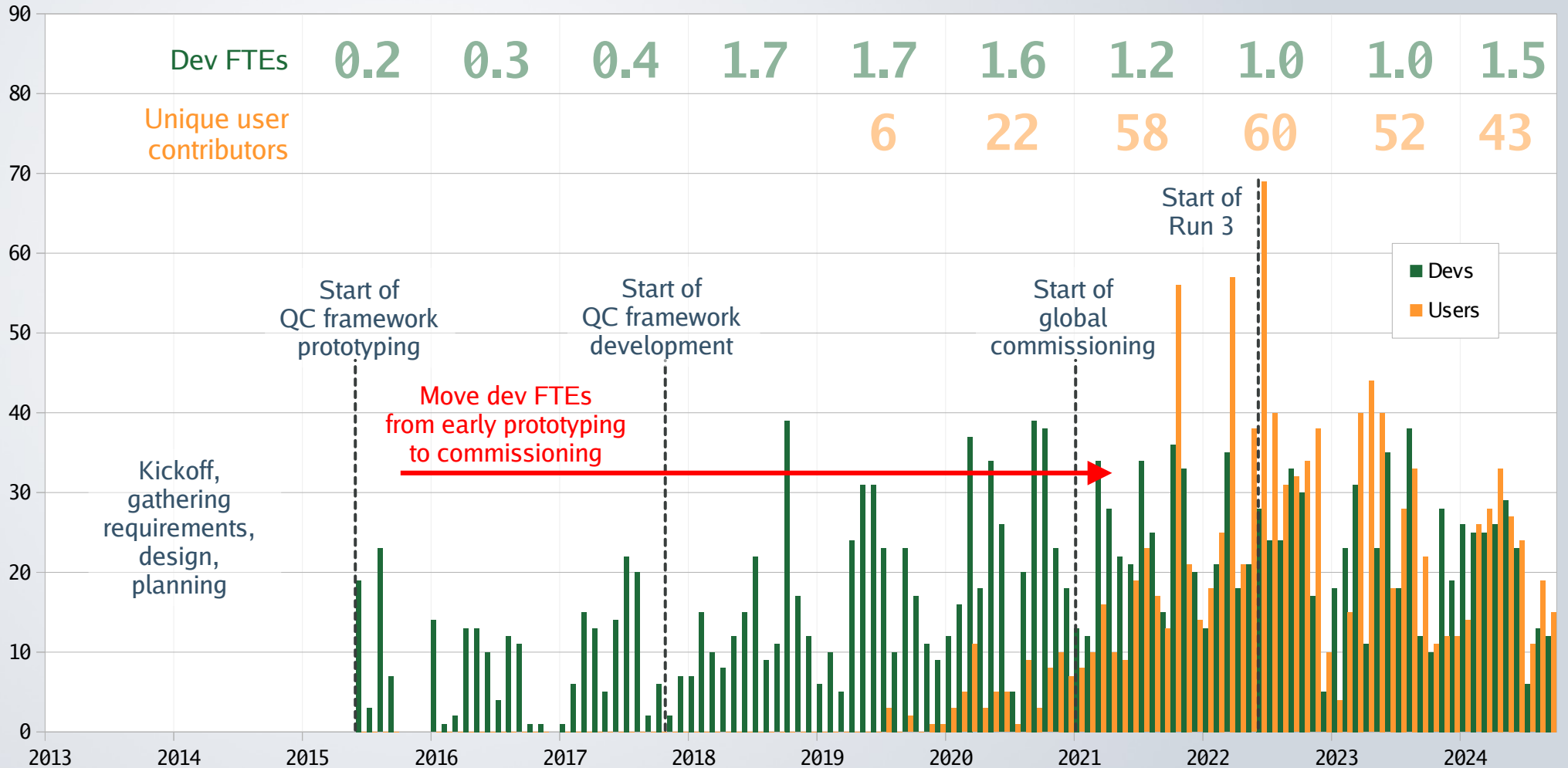
Features not used – correlation



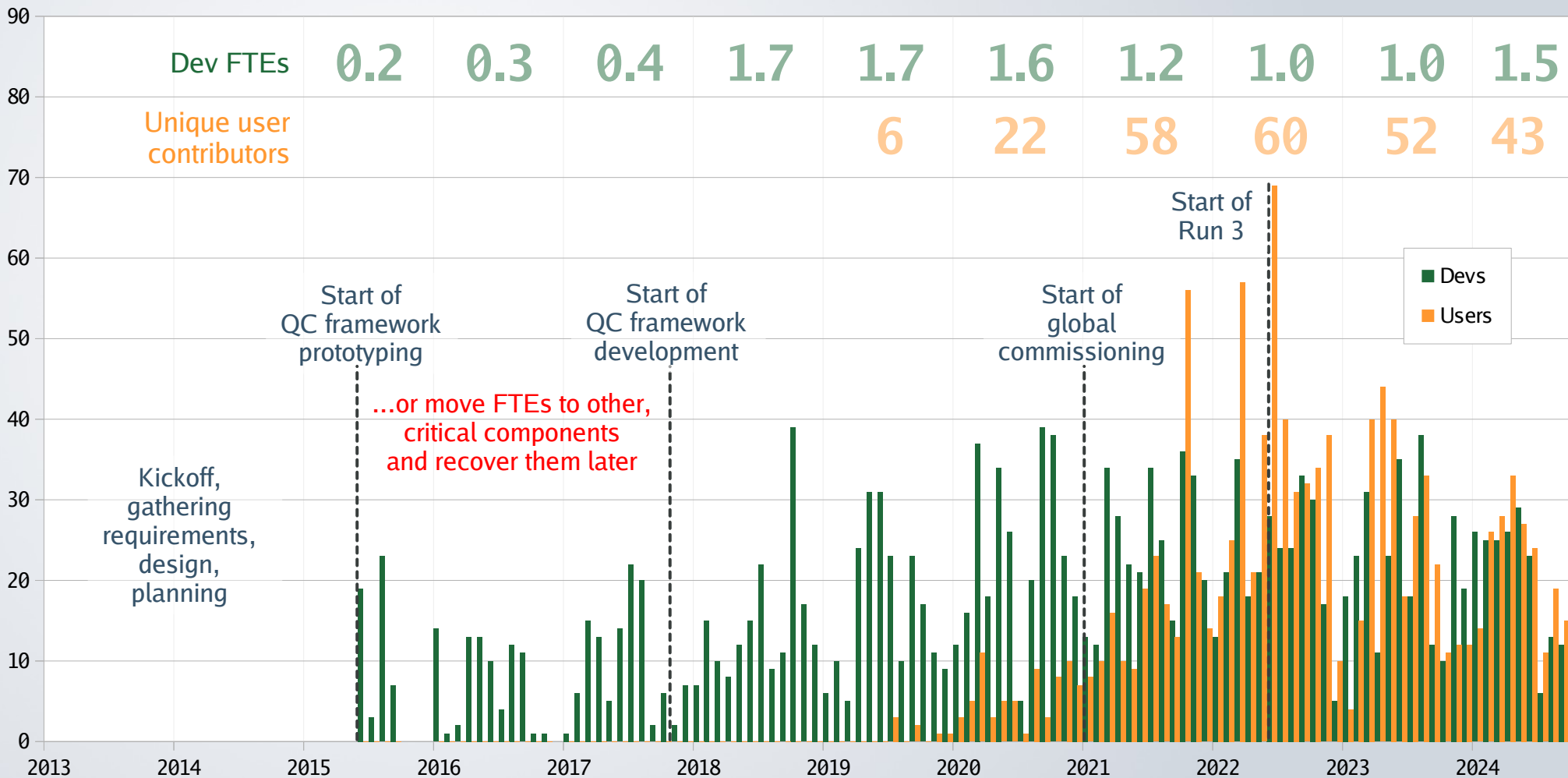
Not done (yet!) – Machine Learning



Possible readjustments to manpower



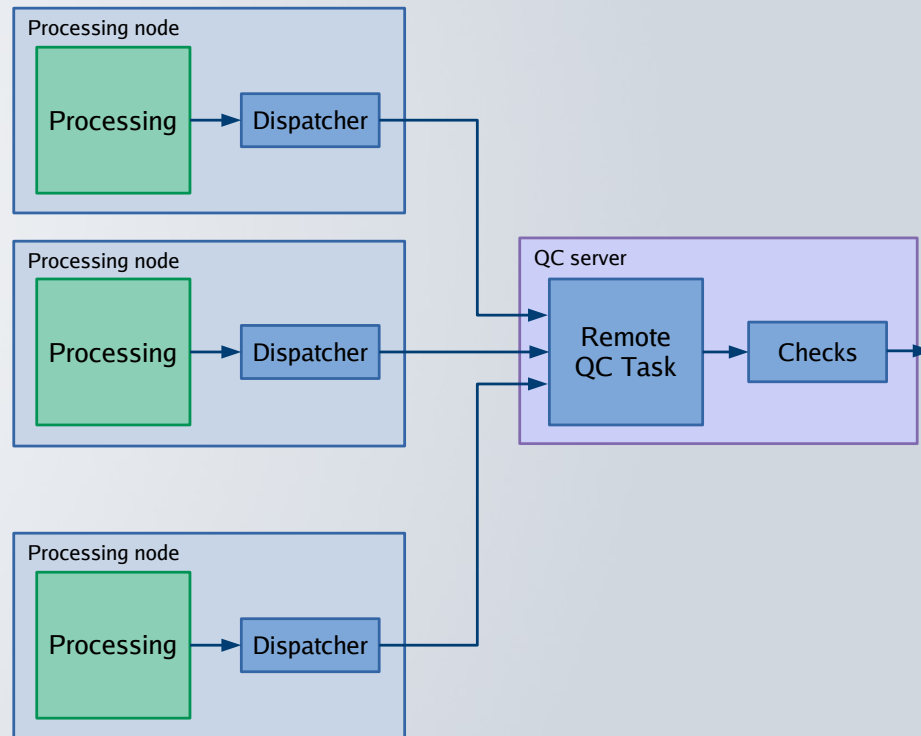
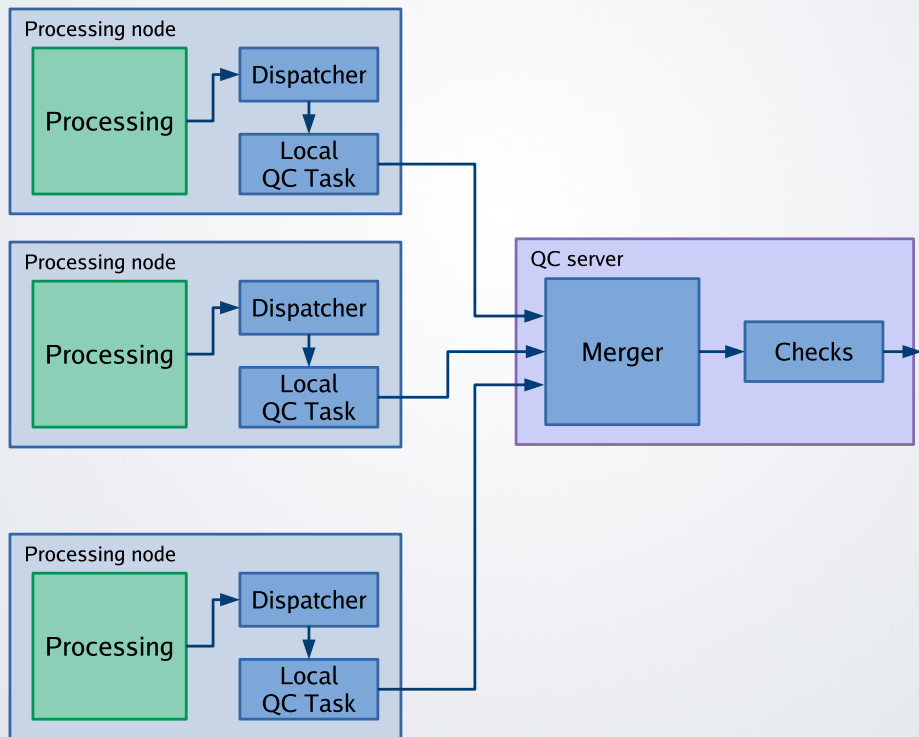
Possible readjustments to manpower



Main takeaways

- When developing a software framework...
 - Assume that performance requirements might always change
 - Take advantage of feedback from early adopters
 - Build a community of users
- When this framework is a component among others...
 - Don't build on top of dependencies which are not there yet
 - Ensure that manpower can be moved between components

Backup slides



Done late – time-based quality flagging

DET A quality:



DET B quality:

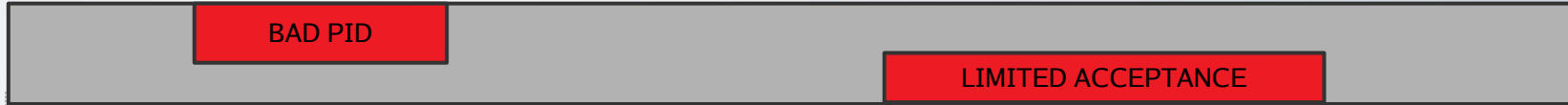


t_{SOR}

t_{EOR}

Done late – time-based quality flagging

DET A quality:



DET B quality:



Good data



Good data incl. Limited Acceptance



t_{SOR}

t_{EOR}

Context switches throughout a working day (April 2022)

