

# ATLAS TDAQ Controls and Configuration software: Evolution from Run 2 to Run 3

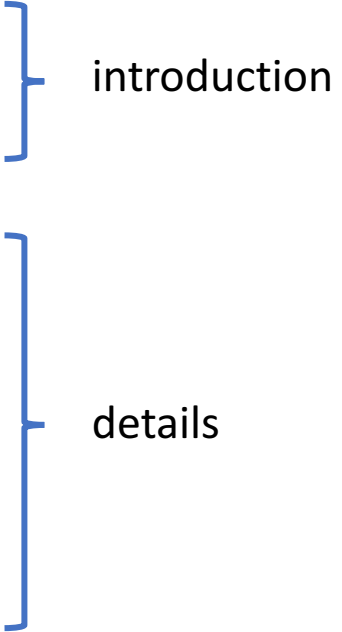
Andrei Kazarov / NRC “Kurchatov Institute” – Petersburg NPI

on behalf of ATLAS TDAQ Controls and Configuration team

virtual CHEP 2021 May 17-21

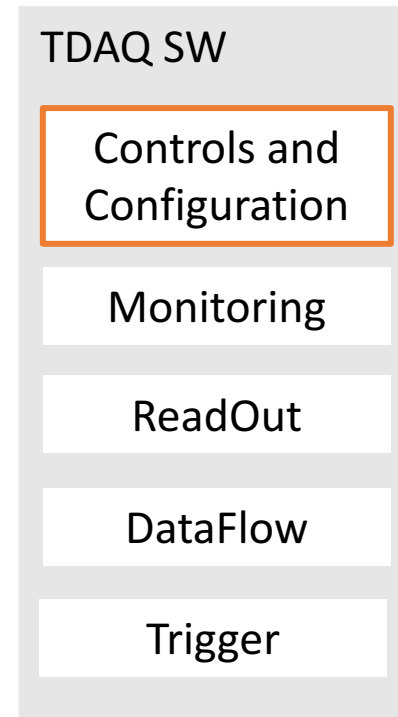


# Content / Overview

- Controls and Configuration software in context of ATLAS TDAQ
  - Software evolution and challenges
  - Run-2 to Run-3 evolution (in few selected examples)
    - Configuration database backend: *upgrade of technology*
    - TDAQ Run Control in a web browser: *new requirements, new package*
    - Expert System: *performance upgrade*
    - Electronic LogBook: *re-vamp of the package*
- 
- introduction
- details

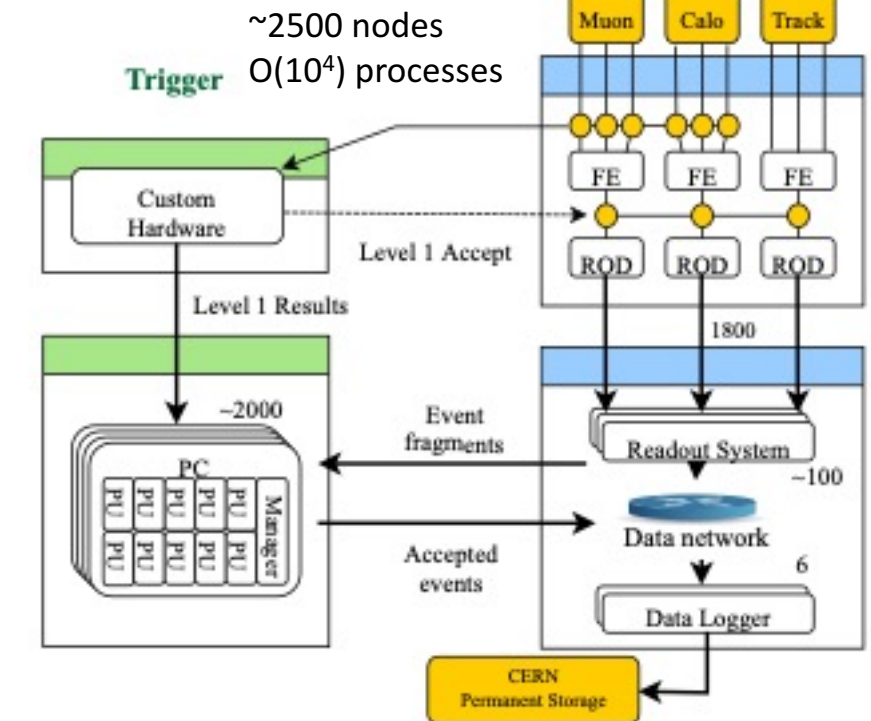
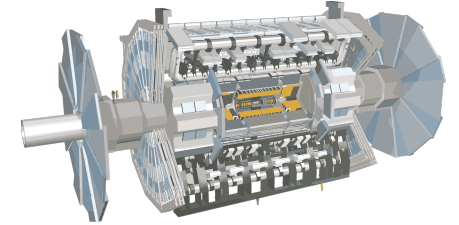
# C&C s/w as part of ATLAS Trigger-DAQ

- Controls and Configuration (C&C) s/w packages are part of a bigger **TDAQ s/w**, including also Monitoring, DataFlow, ReadOut, Trigger steering and algorithms
- C&C components essentially provide a **glue** for all other TDAQ s/w components including Detector readout s/w, Detector Control
  - **Configuration** service
  - Run **Control** framework
  - Knowledge-base tools
  - Web applications for TDAQ operations
- TDAQ s/w is organized in packages (200+), regularly released in form of stable binary distributions



run data-taking

maximize efficiency



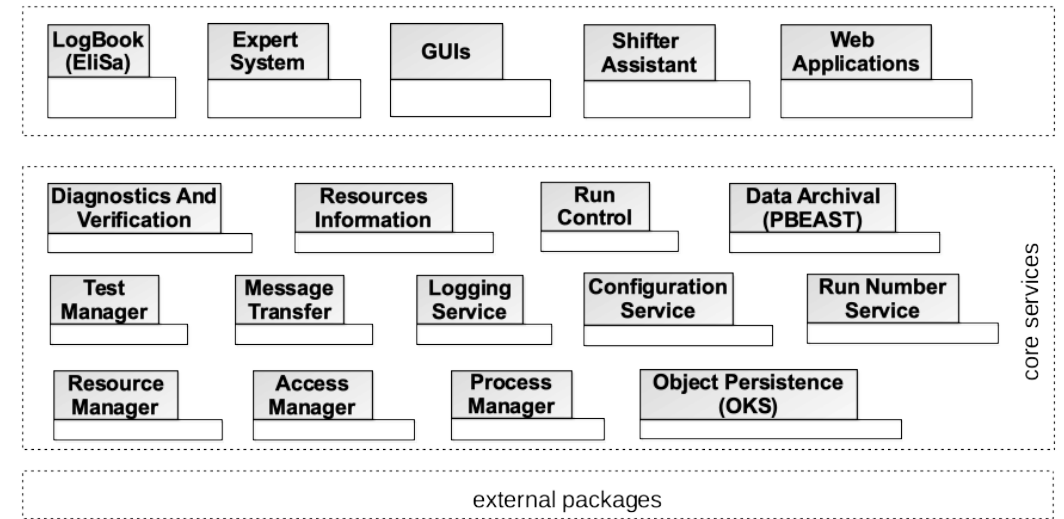
# Evolution and challenges

- Modular, loosely-coupled package-based design, well-established APIs: allows long-term development, evolution and maintenance by a small and non-stable team of developers
- Some architectural and design ideas persist since late '90th
- Must follow LHC run/shutdown cycles, maintaining **gradual evolution** in 1-2 years of shutdown and **stable operations** in 2-3 years of Run periods
- Must follow and profit from emerging software technologies, standards and 3<sup>rd</sup> party s/w
- In every cycle, a few packages undergo a major re-implementation, new packages added, some are dropped/replaced
- Aiming for **simplification** and **modernization** of code, implementation of **new requirements**

[MOR97] G. Mornacchi et al, "The ATLAS DAQ and event filter prototype "-1" project", 10th IEEE Real Time Conference, Beaune, France, 1997

[JON97] Robert Jones et al., "The OKS in-memory persistent object manager", *IEEE Transactions on Nuclear Science, Vol.45, no.4, August 1998*, pp 1958-1964

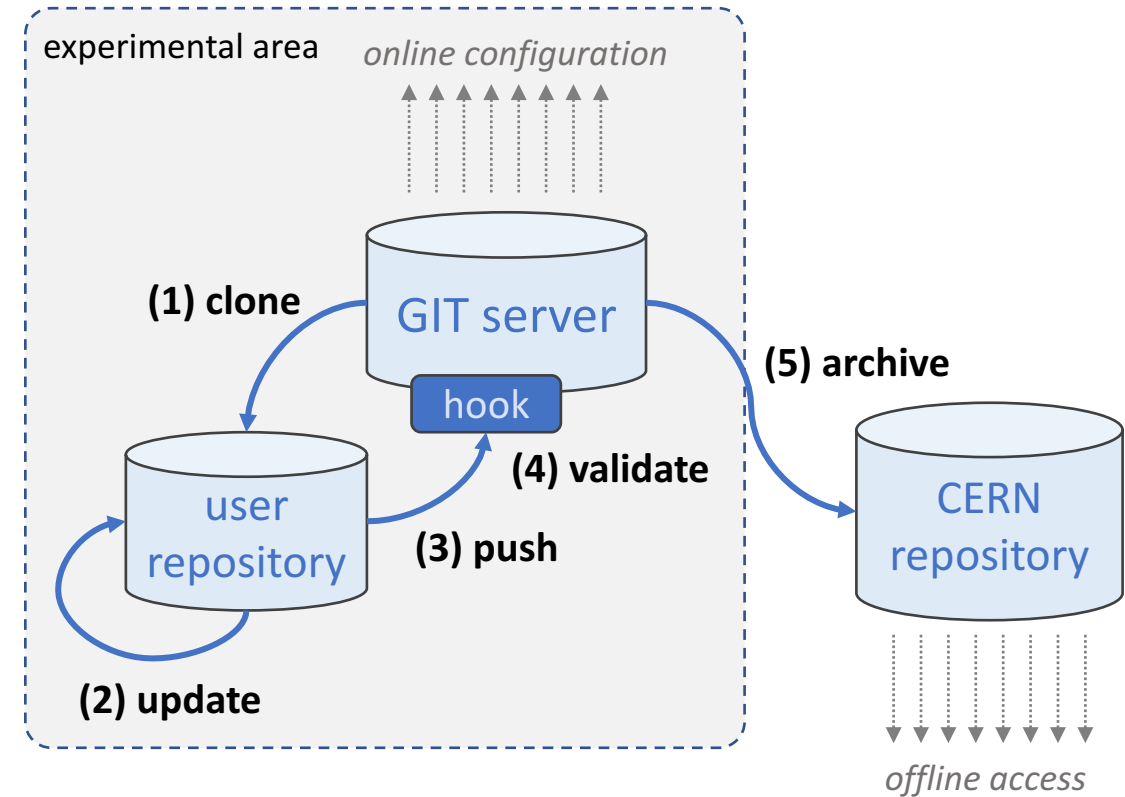
A high-level architecture of C&C software (see backup)



**Run1->Run2 evolution** in more details: A. Kazarov, I. Aleksandrov, G. Avolio, M. Caprini, A. Chitan, A.C. Radu, A. Kazymov, G.L. Miotto, M. Mineev, A. Santos et al., *Journal of Physics: Conference Series* 1525, 012036 (2020) <https://doi.org/10.1088/1742-6596/1525/1/012036>

# Configuration service: CVS to GIT migration

- TDAQ configuration data is stored in +1000 nested XML files updated by many experts
- **CVS** was used to keep the files in a consistent state during Runs 1&2
  - not supported for many years, misses important security and interface improvements
- Replaced by **GIT** solving several issues and keeping design mostly unchanged:
  - Validation of changes on update in server hook: check consistency and access permissions using role-based Access Manager
  - preserve and archive a data-taking configuration by a commit hash
  - expose GIT interface directly to users, including web editing capabilities



# OKS GIT implementation details

- **gitea** (GIT) server on a VM
  - more than 3K user accounts
  - web interface (browse, edit, merge)
  - 5 seconds per commit (including validation)
  - synchronized with central CERN Git service (gitlab) for archiving and world-wide read access
- A configuration **update** integrated with data TDAQ taking session tools:
  - show available **changes as commits** with details to the Operator
- Implemented a workflow on top of GIT merge/pull requests to facilitate scheduled configuration changes during TDAQ operations

The screenshot shows the Gitea web interface for a repository named 'oks / tdaq-09-02-01'. The top navigation bar includes 'Dashboard', 'Issues', 'Pull Requests', 'Milestones', 'Explore', a notification bell, and a user profile icon. Below the navigation bar, the repository name is displayed along with 'Watch 0', 'Star 0', and 'Fork 0' buttons. A secondary navigation bar contains 'Code', 'Issues 0', 'Pull Requests 0', 'Releases 2.5k', 'Wiki', 'Activity', and 'Settings'. A summary bar shows '1825 Commits', '10 Branches', and '4.8 MiB'. Below this, there are buttons for 'Branch: master' and 'Commit Graph'. The main content area displays '1825 Commits (master)' with a search input and 'All Branches' checkbox. A table of commits is shown with columns for Author, SHA1, Message, and Date. The first commit is by Francesco S... with SHA1 680fdf82c5, message 'Back to the standard TTC con...', and date '2 hours ago'.

Author	SHA1	Message	Date
Francesco S...	680fdf82c5	Back to the standard TTC con...	2 hours ago

# Web-based Run Control

- Motivation: provide web-based access to the functionality of Integrated GUI: a standalone application integrating interfaces to C&C services for controlling and supervising ATLAS data-taking sessions
- The required functionality
  - connecting to a TDAQ session for control or monitoring
  - presenting hierarchical TDAQ applications tree (see next slide), dynamically updating states of individual items
  - sending RC commands to applications
  - connecting to Error Messaging service to provide real-time error monitoring
  - log files browsing
- Requirements for the technology:
  - its backend part needs to be tightly integrated with main TDAQ services like Run Control, Information Service and Error Reporting
  - the frontend part should offer a rich set of widgets
  - provide good scalability and conservative resource usage, allowing connections for many ( $O(10)$ ) users and serving multiple running TDAQ sessions in parallel
  - support of dynamic and interactive web features like Ajax or Web Sockets
- **Apache WICKET** was chosen for the implementation: a Java-only powerful backend and simple frontend



# Web Run Control in action

The screenshot displays the ATLAS TDAQ web RUN CONTROL interface. At the top, it shows the ATLAS logo, the date 'tdaq-09-02-01', and the user 'Logged in as: ATLAS'. The mode is set to 'DISPLAY' and the control owner is visible. The main interface is divided into several sections:

- Run Control tree:** A hierarchical tree on the left showing the status of various segments. The top-level segment is '[RUNNING] ATLAS' with a rate of 42.86. Below it are 'Online Segment' and '[RUNNING] TDAQ'. Under TDAQ, there are several sub-segments including 'infrastructure', 'ddcdtATLAS\_ATLGCSDDC', 'L1CentralTrigger', 'HLT', 'TRP\_Segment', 'TDAQ\_Monitoring', 'MUCalServerSegment', 'InnerDetectors', and 'TRT\_Segment'. The 'InnerDetectors' and 'TRT\_Segment' segments show rates of 1.06.
- Rates plots (configurable):** A bar chart on the right showing rates in Hz and busy fraction in % over time. The x-axis ranges from 14:08 to 14:18. The y-axis for rates is logarithmic, ranging from 0 to 100,000 Hz. The busy fraction is shown on a linear scale from 0 to 100%. The plot shows a significant increase in rates and busy fraction starting around 14:14. Specific data points are labeled: 47,437.3 Hz (Recording Rate), 23,760.3 Hz (HLT Rate), and 17,850.3 Hz (L1 Rate).
- ERS controls:** A section below the rates plot with a search filter 'sev=FATAL OR sev=ERROR OR sev=WARNING', a row count of 10, and buttons for 'Pause' and 'Export'.
- ERS messages:** A list of error messages at the bottom, all showing 'WARNING' status and 'rc::ApplicationExited' for the 'TRP\_Segment\_XMON' component. The messages indicate that the application 'xmonadapter\_app' or 'xmonadapter\_app\_mig' exited with code '127' on host 'pc-tdq-mon-07.cern.ch'.

Run Control tree

Run Information

Rates plots (configurable)

ERS controls

ERS messages



# An Expert System (CHIP)

*TDAQ system largely deterministic → Possible to identify “signatures” and react properly*

- CHIP: The **C**entral **H**int and **I**nformation **P**rocessor (aka Expert System)
- An “*intelligent*” application having a global view of the TDAQ system and **taking operational decisions**
  - Handles abnormal conditions (correctly disabling a failing readout channel)
  - Automates complex procedures (reacts on “Stable Beam” condition)
  - Performs advanced recoveries (restarting a Trigger PU)
- CHIP embeds the Java-based ESPER **Complex Event Processing (CEP)** engine
- Large Knowledge Base (KB)
  - More than **300** detection “rules” or “directives” expressed and stored in EPL

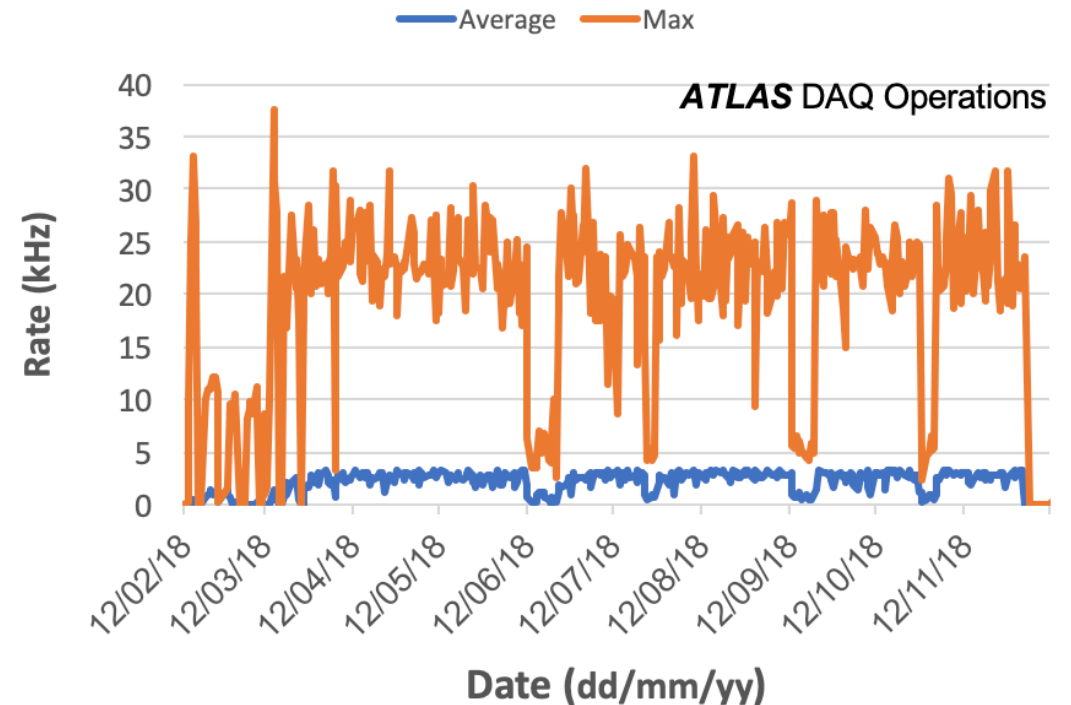


# CHIP performance

- Performance in Run 2
  - Single CHIP instance adequate to monitor the whole TDAQ system
  - Data injection peaks into the ESPER engine up to **40 kHz**
  - Average execution time of KB rules of about  $2 \mu\text{s}^*$
- From Run 2 to Run 3
  - Extension of the KB (13% increase in number of rules)
    - Including new scenarios and integrating new TDAQ components and systems, e.g.
      - Integration of the new readout system SWROD (including stop-less removal and recovery)
      - Restarting of the RootController (egg/chicken problem)
  - Move to Esper 8
    - Knowledge base compiled in Java bit code
    - Evaluation of rules reduced up to **40%**

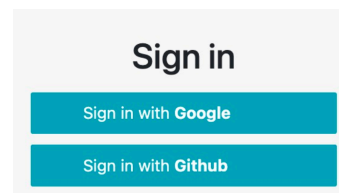
\* Running on dual-socket Intel Xeon E5-2680 V3 - 64 GB RAM

## Event Injection Rate into the ESPER Engine



# Electronic logbook (ELisA) re-vamp for Run 3

- for Run 2 we developed a complete s/w suite to cover an electronic logbook services: web interface, REST API, user authentication, database storage of messages and configuration, client API libraries and utilities, configurable email notification, mailer client
- Reasons for a change:
  - SW updates: the code was ~8 years old
  - Maintenance improvements: two different applications handling web interface and REST API requests, and separate Tomcat web server
  - Deployment improvements: more clients interested to set up their own standalone logbook
  - Requests for new features
- Getting ready for Run 3:
  - Simplification and refactoring: merged the web interface and REST API applications, + embedded web server
  - The application packed and distributed as an RPM, and managed automatically as a service. Custom client configuration is preserved.
  - Support for MySQL backend in addition to Oracle
  - Social login
- The solution is spreading: currently ELisA is used by 14 different clients, half of them beyond ATLAS



# Conclusions

- We provide a stable stack of software for smooth steering of the TDAQ data-taking, for all ATLAS development and operational periods started in early 2000, ranging from core system-level services to web-based applications
- Complexity of the system and need in high data-taking efficiency requires s/w solutions which can implement a high level of operational automation
- Maintaining the high quality of s/w through 20-30 years of the experiment lifetime requires its permanent and gradual evolution, including use of new s/w technologies and partial re-development of the components

# Backup

# Run 2 – Run 3 evolution

- Configuration database backend: from CVS to GIT
- Web-base Run Control: full remote control of data-taking in a browser window
- Expert System: focus on performance
- Electronic LogBook: re-vamp
- Evaluation of a time-series DB for a persistency backend for operational monitoring archival
- Dismiss a very old package CLIPS, replaced by in-house rules engine

