APPLICATIONS OF EXPERT SYSTEM TECHNOLOGY IN THE ATLAS TDAQ CONTROLS FRAMEWORK

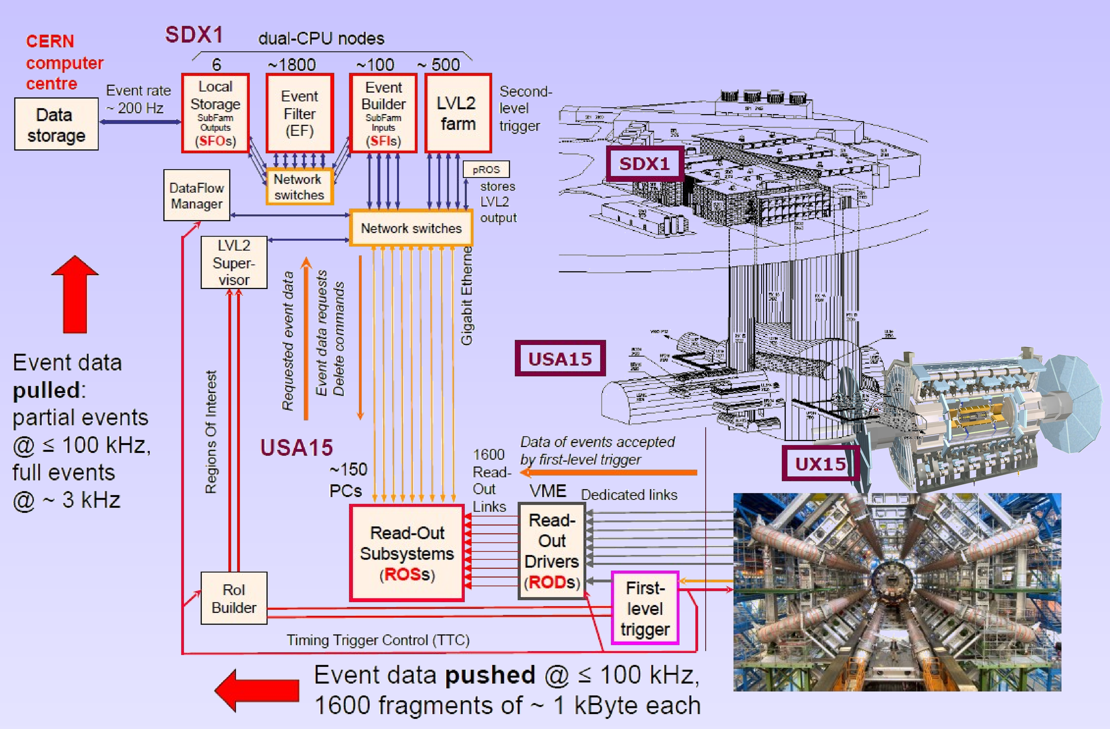
A. Corso-Radu[1], A. Kazarov[2,3], R. Murillo Garcia[1], G. Lehmann Miotto[2], L. Magnoni[2], J-E. Sloper[2]

[1]University of California, Irvine, USA
[2]CERN, Geneva, Switzerland
[3]PNPI, St. Petersburg, Russian Federation

1. Trigger and Data Acquisition of the ATLAS experiment at LHC

ATLAS Experiment
A Toroidal LHC ApparatuS (ATLAS) is a particle physics experiment at the Large Hadron Collider (LHC) at CERN.
The LHC is producing proton-proton head-on collisions with center-of-mass energy equal to 7TeV (design energy 14 TeV) at 40 MHz rate.
The ATLAS detector comprises more than 140 million electronic channels.

Trigger and Data Acquisition system
managing filtering and transfer of ATLAS experiment data from the detector to mass-storage
large-scale, distributed computing system: 10000 applications (30000 is foreseen)
running on 1500 nodes (3000 in near future)
non-stop operations 24hrs/day, 7 days/week in next years



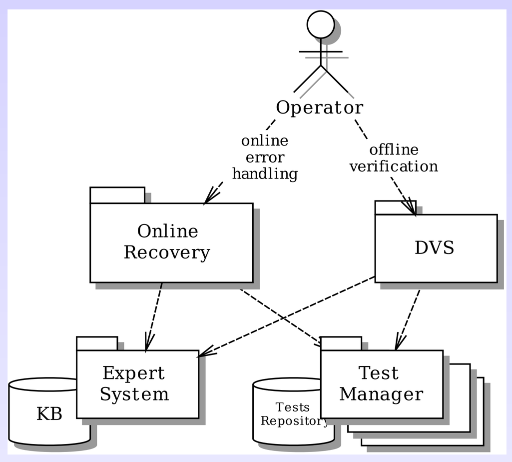
2. Controls framework for TDAQ system

Main objectives
•control and supervise all ATLAS TDAQ h/w and s/w:
– Up to 3000 computer nodes, including SBC, ROSes, rack-mounted PCs (x4, x8, x16... cores), file-servers, monitoring and desktop PCs
– 10000 – 30000 applications
•Provide testing and verification functionality to detect possible errors at early stages
•Handle run-time errors and suggest recovery solutions
•Perform automatic recovery actions
•Keep the expertise of system developers through years of experiment lifetime

Runtime failures are frequent: our goal is to minimize downtime of the system!

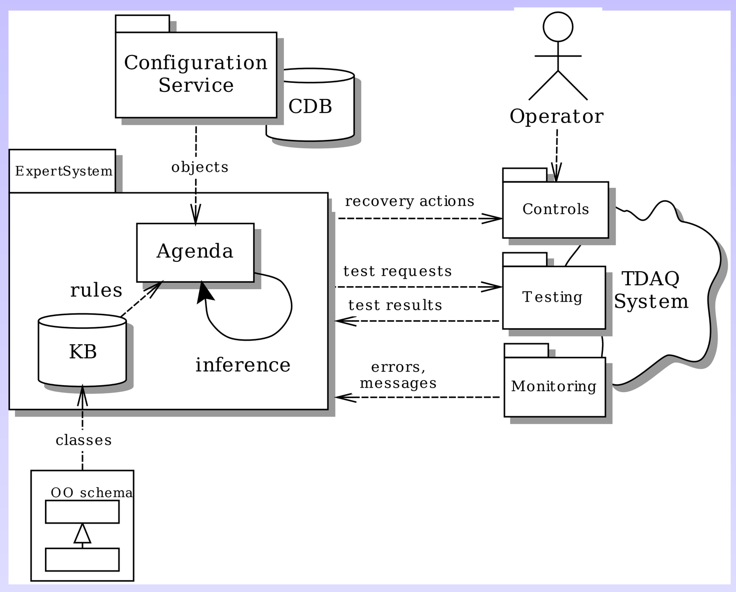
Architecture principles
•System is built of a number services and components, interacting with each other
•Framework-based approach: functionality is extended by users and subsystem experts
•Configurability: the current h/w and s/w state of the TDAQ is stored in the database, all components are configured
•Use of expert system technology for automation of control actions

To accomplish its objective, the Controls system includes some high-level components which are responsible for the automation of system verification, diagnostics of failures and recovery procedures: DVS (Diagnostics and Verification System) and Online Recovery. These components are built on top of a common technology of a forward-chaining Expert System (ES) framework, that allows to program the behavior of a system in terms of "*if-then*" rules and to easily extend or modify the knowledge base (KB).



3. Integration of expert system to the Controls framework

The whole TDAQ system is fully described and configured via the Configuration Database Service according to a pre-defined configuration object-oriented schema. The corresponding schema developed in the CLIPS object language COOL is loaded to all instances of the ES. The actual set of objects for a particular configuration is loaded into the ES as a class hierarchy representing proxies of the applications and the hardware in the system. These objects, together with information gathered from the Monitoring service and test results can then be used in the ES engine to match the loaded rules. The CLIPS inference engine adopts a forward chaining approach: the agenda is how CLIPS keeps track of which rules are to be executed next run, and a rule is added to the agenda when all its conditions, given the status of objects in the systems, are satisfied.



CLIPS parses the knowledge base at run time. This allows to easily customize the behavior of recovery procedures supplying different set of rules as arguments to the recovery applications. In a complex and dynamic framework such as the TDAQ system it is very difficult to detect apriori all the different errors that might occur and which appropriate actions should be taken. It is therefore very important that the expert system can be easily changed and customized as more data is gathered and a better understanding of the system is gained.

4. CLIPS expert system shell

After an evaluation, a open-source product was selected as the core of the implementation
For expert-system based applications.

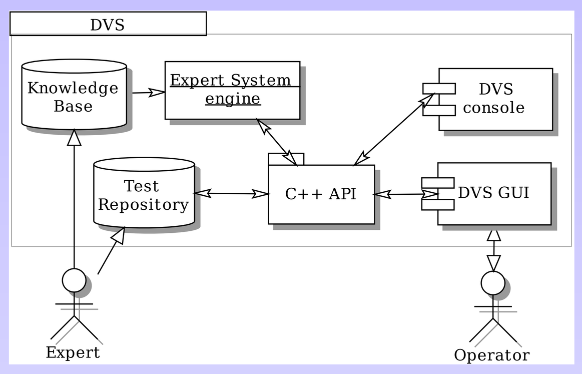
CLIPS = **C** **L**anguage **I**ntegrated **P**roduction **S**ystem, which means
- rule-based ("if-then" paradigm known as "production" system) expert system engine
Open source, originally developed by NASA
can be used as a standalone application or as embeddable (C-source) library, which allows integration of ES functionality in the existing applications
implements a number of different programming paradigms:
– "if-then" rules and a forward-chaining inference engine
– object-oriented constructs ("COOL" language)
– traditional algorithmic constructs (functions, loops etc.)

One of the key (if not unique) features of CLIPS is the possibility to describe the architecture of your application using an object-oriented approach (including classes, multiple inheritance, methods) and to develop "generic" rules for classes of objects which will be then applied to a particular set of objects according to a particular configuration without any modification of the pre-loaded rules. This model fits very well with the TDAQ configuration object-oriented approach and gives a powerful tool for developing rule-based control applications in TDAQ framework.

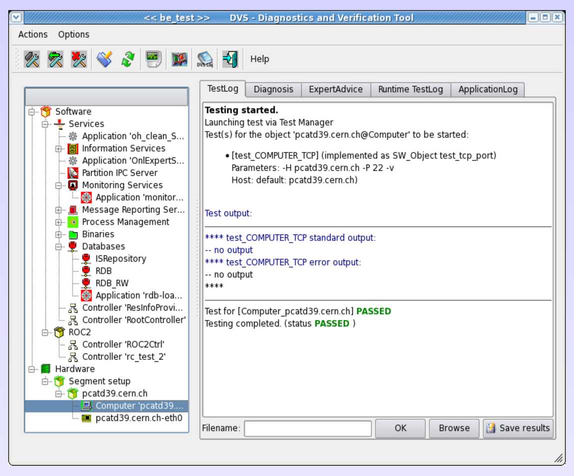
5. Diagnostics and Verification framework

DVS (Diagnostics and Verification system) is a framework used to assess the correct functionality of the system, to detect and diagnose eventual problems. DVS allows the configuration of one or several tests for any component in the system by means of a configuration database. The system and the testing results can be viewed in a tree like structure using a user friendly graphical user interface.

Test Repository service allows configuration and execution of tests for single TDAQ component. A test is a binary running on a particular host in a system. The following attributes are available for test description: *parameters*, *host*, *dependencies*, *time-outs*, *scope*, *complexity* and *mask*. Any test can be associated either to a particular object in the configuration database or to all objects of a particular type.



Using **DVS G**raphical **U**ser **I**nterface (right hand side) The user can select a single component or a group in the tree and run all defined tests by clicking a single button. As tests finish, the components icons change colour reflecting the result. On the right hand side the test results, diagnosis and recovery advice are presented.



Other features were implemented following the constant user feedback received: log file browser for accessing log files produced by the TDAQ applications running in a distributed environment, test scope to prevent destructive tests to be executed during data-taking sessions, test runtime output for long-running tests.

6. Online Recovery scenarios

Code on the right describes a general recovery mechanism in case an application has died. The left hand side (LHS) of the rule contains the conditions to identify the error situation, the right hand side (RHS) lists the actions to be taken.

```
if
system state is running, and
application App1 status is absent, and
application App1 has supervisor S1, and
application App1 membership in
then
notify S1 ignore App1
notify controller ignore App1
set membership App1 out
```

In the TDAQ framework applications are arranged in a tree structure, with a controller responsible for a segment, a subset of the whole system. Following this recovery rule, when an application that satisfies the conditions described in the LHS the expert system takes care of notifying the associated controller to ignore the application from now on.

The expert system server holds the main responsibility of dealing with detector frontend failures. There is an automatic recovery procedure which allows for Read-Out Drivers (RODs) that are permanently busy or otherwise faulty to be recovered without stopping the run. The expert system drives the recovery operations, detecting the error conditions and holding the data acquisition triggers for the time needed to disable the failing component and to restore safe data taking operations.

Using ES rules it's possible to automate some routine actions normally performed by hand by system's operator on shift. The ES rule presented on the right side is an example of such action. System automatically resumes the trigger when collider energy reaches 2.7TeV during the ramp.

```
(defrule resume-trigger-on-energy
  (selection ?val&~BCref)
  ?fact<-(trigger-on-hold)
  (not (clock-changing ?changeFrom))
  (object (is-a IS-INFO)(IS-NAME RunCtrl.RootController) (FIELD state)
(VALUE RUNNING))
  (object (is-a IS-INFO)(IS-NAME LHC.BeamMode) (FIELD value)(VALUE ?
bm&PREPARERAMP|RAMP) )
  (object (is-a IS-INFO) (IS-NAME LHC.BeamEnergy) (FIELD value) (VALUE ?
energy))
  (test (> ?energy 2700))

  =>

  (ers-warning OnlRec::EndRampPause Resuming the trigger as the Beam has
reached and energy of 2.7 TeV.)
  (cmd-trigger-go)
  (cmd-send-sync RootController LUMIBLOCKINCREASE)
  (retract ?fact)
                                    )
```

Conclusions and Outlook

A rule-based expert system is used to help the TDAQ operators to control data-taking of the ATLAS experiment:
• we are capable of restarting processes on the fly
• we are capable of excluding/re-including parts of the readout dynamically
• we are capable of re-synchronizing detectors dynamically
• we are interacting with the accelerator status to automatically select the correct clock and preserve the DAQ during changing LHC conditions

More and more expert actions are being coded into the expert system thus allowing already today (1st year of steady data taking with LHC) to operate the TDAQ with non-experts.
We are envisaging an extension of the usage of expert knowledge to introduce a new component, the Shifter Assistant, that will advise and guide the operator in different situations.