SOFTWARE AND CONTROL ISSUES

Delphine Jacquet BE/OP /LHC, CERN, Geneva, Switzerland

Abstract

The software applications and fixed displays in the control room are the unique windows to the LHC, the interface used to give it orders, diagnose its state of health and control its behavior. The better tools we have to communicate, the more efficient is the operation team to detect and cure problems, and also run the accelerator in an efficient way. Despite the impressive number of well working applications available in CCC, there is still room for improvement. This paper describes the main difficulties and issues encountered during 2010 LHC run that could be solved by improving the existing software applications, or by creating new ones.

INTRODUCTION

2010 has been a year of debugging for software in all domains of the accelerator and at each layer of the control system, from the PLCs to the user applications. The debugging and solutions of the diverse issues has been done with an amazing reactivity from the equipments control experts and high level application developers. They had to be very flexible to cope with the fast evolution of the LHC and to accept new requirements that came up. At the end of 2010 run, we can be proud of the impressive number of well working applications that are available in the CCC. At the same time, a full year of experience with the machine operation also leaves us with a big list of things that should be improved to make LHC operations easier and safer. A not exhaustive list of the mains requirements is presented hereafter.

EQUIPEMENT CONTROL

In general, the equipment software is well under control now. Still, some requirements to strengthen or improve the software are expressed.

TCDQ software

During 2010 run, several control problems for the TCDQ have been encountered. The origin of the main problem is the FESA class that does not handle properly the TCDQ statuses. The PLCs, low level control of the TCDQ is designed with the standard BT interface for SPS girders (MST, MSE and ZS), whereas the FESA class has to provide an LHC collimator like interface. In some cases, the mixing of status handling between low level and FESA led to the following problems:

- TCDQ stays armed when it is already at the requested position, then it does not accept other command until manually disarmed.
- TQDQ reports an idle state, but in reality stays armed and then moves unexpectedly at the first collimator timing event.

These problems appeared in the middle of the run in case of some combinations of expert commands, whereas it was not seen for standard operation.

As a solution, Etienne Carlier's team has already developed a new version of the PLC software that will handle by itself all the statuses and provide an LHC collimator like interface. The FESA class will be simplified, and will only publish the low-level statuses. This has been tested already in the test bench and will be deployed during the shutdown.

In addition, Etienne Carlier recommends creating separate sequences for the TCDQs that can be better adapted to its particularity.

RF

The control room applications for RF systems cover the needs to control and drive the RF cavities and ADT.

Nevertheless, RF diagnostics tools are still missing, like a detailed panel of the hardware interlocks for the RF lines, and a better display of the RF signals (e.g. mountain range) in the CCC.

Power converter PIC and QPS

In the control room there is currently no efficient way to restart individual power converters after a trip. Global sequences to prepare or drive the power converters by sector are available, but if we need to control only a sub-set of them, not less than 4 applications are needed:

- The equip state application to reset and drive to the operational value
- The PIC application to reset and give permit
- The Circuit synoptic application in case QPS switches need to be closed
- The generation application to set resident the necessary beam processes.

We have to switch from one application to the other to perform resets, open switches, give permits, reset again and drive to the right settings. In addition, the PIC and circuit synoptic application requires a special password as these systems are not protected by RBAC. This procedure needs to be simplified, for example with a dedicated sequence.

SOFTWARE FOR INJECTION

As shown in the presentation given by Stefano Redaelli about the LHC turn over [1], the injection process is very time consuming compared with the other phases of operation. As opposed of the pre-cycle or ramp phase, whose duration is driven by the magnet functions, it is possible to reduce the time spent at injection with some software modifications. At the same time, the process to prepare the injection scheme can be more efficient and the risk of getting a wrong circulating bunch configuration can be reduced.

Injection Sequencer and IQC efficiency

As the injection sequencer is designed, the B1 and B2 injections are done one after the other. This implies that to start requesting injection for one beam, the IQC analysis of the other beam has to be finished. As the IQC analysis takes time, the supercycle length of the SPS has to be adapted to achieve one injection every supercycle, but this is difficult because the analysis time of the IQC is not constant. Therefore, the injection sequencer should be modified to make B2 injection possible as soon as B1 is injected, i.e. without waiting for the IQC analysis. This would allow for shorter SPS supercycles with always optimal length. This change is limited to the injection sequencer GUI: no change has to be done at the IQC, SIS or CBCM level.

The other time consuming factor is the many IQC latches. The IQC is there to give a qualitative result of the injection, and the chosen design was to stop the injection process if the quality of the injection is not optimum. The problem is that several iterations were needed to estimate the correct thresholds (MKI pulse or BLM thresholds) needed to fairly qualify an injection as “good” or “bad”. In addition an IQC latch didn’t lead to any special corrective action, the OP team was instructed to unlatch and continue. One possibility to improve the situation would be to relax the threshold so that the IQC doesn’t latch too often, but then we would miss the valuable information that the quality of the injection wasn’t optimal. An other idea to consider is that “good” or “bad” is not enough, and an intermediate level could be added meaning that the process do not stop when the quality is not optimal, but the status is given as an information to the operation team. The level “bad” has to be reserved when an injection is so dirty that an immediate action has to be taken.

Also to be improved is the IQC playback application that should allow analysing the injections quality offline. For example, it misses filters by injection result.

Circulating bunch configuration

The circulating bunch configuration for each beam is an array with all the RF buckets that are filled with a bunch. This important information is distributed to the experiments via DIP and to certain equipment and software via a FESA class. It is also used by the injection sequencer to prevent unwanted over-injections. It is then very important to get it always right.

The circulating bunch configuration is updated by the injection sequencer according to the IQC analysis, which is responsible to publish the injection result. The decision of the IQC that beam has been injected or not relies on 2 BCTs measurement in the extraction line and the kicker pulse. Whereas it worked well with protons, with ions the BCT started to give false data to the IQC, which then reported an incorrect status of the injection (beam in when no injection has actually been done, or no beam even if the injection was successful). This implied that the circulating bunch configuration wasn’t updated properly, with the consequence that the beam couldn’t be injected anymore because the requested bucket was seen as filled, or we had a risk of over-injection because the injection sequencer repeated the request on the same bucket thinking it was empty. When it happened, the only possibility to go on with injection is to update the circulating bunch configuration table directly in the database, with the implied danger of database manual updates.

To reduce the risk of over-injection, soon a check has been added in the injection sequencer: before each injection the circulating bunch configuration is compared with the measurement given by the BQM, and a warning is given if this is inconsistent. The database is then corrected if needed.

In addition, the IQC should cross check the transfer line measurement with a ring measurement like a ring BCT or the LHC BQM.

It is already foreseen to add functionality in the LHC BQM to update the circulating bunch configuration directly from the filled bucket measurement, so no more manual update of the database would be needed in case of problem.

Filling schemes

For the 2010 run, almost 100 filling scheme have been created. The existing software to insert this filling scheme in the database is not flexible and efficient enough.

The bunch patterns, which are the SPS beam description that is then used to get the right bunch configuration in the LHC, couldn’t be created with an application but needed a direct update of the database by an SQL script. As a consequence, it is not possible to create new pattern without the knowledge of the database design and the connection right to the LSA database, and this has showed to be too restrictive especially during

MDs. A panel will be added to the injection scheme editor to create new bunch patterns.

The filling scheme had to be created manually, first creating the injection requests then assemble them. The source information to create a given filling scheme is a text file given by the machine coordinator containing description of all the necessary injection. The injection scheme editor application should be improved with the functionality to create the complete scheme directly from the text file. In addition of being much more efficient, this would also reduce the risk of errors.

LHC SEQUENCER GUI, SEQUENCER EDITOR

LHC sequencer GUI

The LHC sequencer is the key application in the control room, it has to be intuitive, easy to use, clear and above all, very safe.[2]

The first GUI that was developed by OP was not satisfactory. Users lost trust in it after experiencing some dangerous issues like the "run through" bug (the sequencer continuing to execute the following tasks even if the user pressed "step" to execute one task only). This GUI also had some layout problems. Therefore, it was decided to replace it by a new GUI developed by CO/AP section. The sequencer server was not changed. The good things of the previous GUI, like the quick launch panel, have been kept, and all the other OP requirements implemented. Still some improvements of the check list panel are needed. The sequencer GUI could also improve its flexibility by set the tasks parameters for certain tasks. (It would be useful if we get a sequence to restart a given power converter).

Sequencer Editor

During the 2010 run, an impressive number of tasks, sub-sequences and sequences were created. The existing software, the sequence editor, should be reviewed to include the listed requirements:

- A subsequence should be independent of a sequence, now a lot of sequences have been created with the only aim to contain the sub-sequences.
- The GUI should allow to copy, cut and past tasks, sequences and sub-sequences, possibly using drag-and-drop interactions.
- A clear tasks and sequence catalogue should be available.
- Change history should be made available for tasks, sequences or sub-sequences to show who changed what and why. A rollback possibility would be appreciated as well.

A new database schema is being implemented which covers part of the requirements listed above. A new sequences editor will be developed in 2011 to include all the new requirements and implement the new database schema.

LHC nominal sequence

The nominal sequence contains all sub-sequences and tasks needed to drive the LHC from ramp down to collisions. It has changed a lot during the run, following the fast evolution of the LHC: tasks have been added to replace manual actions, or to solve some issues. Others have been discarded or replaced. However, the overall structure of the sequence, especially the sub-sequences to prepare the LHC for injection, should be restructured to allow for more parallelism. For example, creating parallel sub-sequences that act in a single type of equipment would make this phase much more efficient.

The maintenance of the nominal sequence is a collective effort of several members of the operational team, who have to agree on a common way to operate the LHC. Better procedures and tools have to be put in place to prepare changes, to keep track of them and to distribute information about them to all the concerned persons.

STATE MACHINE

A state machine representing the functional states of the accelerator has been defined, together with a list of checks to be executed on each of the transitions. These checks verify for instance that certain actions have been carried out or that all the necessary equipments are ready for the next state. The checks are implemented as tasks in short "check list" sequences.

The state change is driven by the LHC sequencer

- A task in the nominal sequence request a change of state
- The state machine executes the check list sequence
- If all the tests are successful, the change of state is done.

A GUI application displays the state machine diagram and monitors the actual state. Another application shows which of the tests failed or were successful. The two applications will be combined into a single one soon.

The status of the state machine is well advanced; the mechanism for state transition has already been used at the end of the run and will be really operational next start-up. Still, the check lists have to be reviewed and some check tasks added.

In the future, the operational state will be distributed to other software and equipments to enable them to constrain their behaviour depending on the state, with the goal to increase safety of operations. For example, the sequencer will play certain tasks only in the appropriate operational state, and LSA will load certain beam process only in the corresponding operational state.

LSA AND SETTINGS MANAGEMENT

Problems with some hardware functions

Some of the hardware functions generated with LSA have a lot of constraints and end up being quite complicated, a good example being the tune-trim system and the RQTD and RQTF functions.

- Lots of source parameters and associated makerules
- Fast optic change during the squeeze
- Function has to be smooth and continue along the hypercycle (incorporation rules).

The so generated functions may have 2 kinds of problems

- The function doesn't load because rejected by the FGC. The FGC can raise an "Invalid time" exception in case some points of the function are too closed to each other, or a "di/dt out of limits" exception in case the function has at least 2 points with a di/dt over the FGC limit. The invalid time problem has already been solved by adding filters in the makerules. For the di/dt problem, a check of the max di/dt of the function will be added at the makerule, incorporation rule or value generator level. The advantage will be that if the function is not valid, it will be detected immediately and the trim (including regeneration and incorporation trims) will be rejected, whereas actually the problem is seen only at the moment we load the function (and the setting expert potentially long gone home).
- The function is loaded without problem to the FGC, but it trips the power converter as soon as played because the acceleration rate is too high and seen as a quench by the QPS. Implementing a check of the acceleration rate is not obvious, because it is a very difficult parameter to estimate for a function. The current evolution between 2 points has to be interpolated, but it hardly reflects the reality and can lead to reject functions that wouldn't cause real problems. Greg Kruk is working on a suitable solution. At the same time, work has to be done on a better smoothing of the function to avoid spikes. The idea is to add an intermediate parameters called Ksmooth that would handle the multiple sources for RQTF and RQTD, and apply a smoothing already at this level.

Other issues to be addressed

<u>Incorporation</u>

The incorporation has been one of the trickiest settings manipulations this year. This mechanism is quite complicated and difficult to understand for non expert, and this has sometime lead to errors. It has also some limitations that should be addressed:

- The possibility to define many incorporation ranges per beam process is already there, but should be improved with the possibility that a rule defined for a given range can modify the whole beam process.
- The GUI should help the user with the definition of ranges and in and out parameters. (Predefined parameters like start/end of beam-process for example).
- More sophisticated makerules should be created, to deal for example with the snapback or dynamic correction of B3 at injection.

LSA team will review, complete and simplify the incorporation mechanism and clarify the associated GUI .

<u>Traceability, settings rollback and setting check</u>

It would be very useful to have a way of logging every driven parameter and each beam process that is made resident. The trim history that we have is good when you know the parameters that have been trimmed, but if you want to know what parameter has changed it doesn't really help. At the same time, a rollback application (settings recovery at a given time), that would handle compound trims (e.g. orbit trims), is mandatory.

To guaranty the sanity of the settings, especially after MDs, we should be able to compare them with a reference beam process, which the details of implementation are still to be discussed. In a more general way, we should be able to easily compare settings of any beam processes.

<u>MCS (critical settings)</u>

The problem with MCS is that the re-generation of actual beam processes (that is done after ramp and during squeeze) does not work for critical settings as for the other parameters: with the present implementation, one can obtain usable settings only with expert signature (given by RBAC roles). If the expert is not there, in principle, the actual settings for critical parameters can't be regenerated. As this couldn't work in daily operation, complicated work around has been put in place: extra users linked to archived beam process for which the signature was manually generated. This makes the system more complicated and more difficult to maintain.

A solution for this problem is being implemented and will be in place next start-up.

An other issue with MCS is that it is not possible to load a segment of the function, as it is done for example for the squeeze function. As the possibility to have a combined ramp-squeeze-collide beam process is now seriously considered, this problem has to be addressed urgently.

OTHER SYSTEMS

Alarms

The alarm is now a robust and reliable system that could be used in a more efficient way for the LHC:

The LHC alarm screen is permanently filled with red alarms, even when the machine is working perfectly, the effect being that an important alarm can easily be ignored.

- OP and the equipment responsible should review the alarm configuration: what alarms are really needed by OP, and for each of the alarms carefully review the level.
- It would also help a lot if the alarm system where able to handle the machine modes properly, because, for example, an alarm can be very important at injection but completely ignored for other machine modes.

Diamon

When a problem occurs on a given application, it is often difficult for OP team to find the basic information like: what is the associated front end? Which software layer (e.g. middle-tier server, JMS broker, etc) causes the problem? Can the server be restarted without affecting the beam? Etc… Clear information of the dependencies between software processes and layers should be displayed in Damon.

In addition, Diamon does not always display the correct server status: some processes are permanently red whereas some others stay green even in case of problems.

This should be solved to make Diamon really useful for operations

Front-ends and Proxies

Front-ends and proxies still crash too often. This has often annoying effects, like data missing in the logging database, impossibility to perform measurements, loss of communication with experiments etc… And it can sometime have a big impact on the LHC efficiency if front ends of critical systems are affected.

Orbit and tune feedbacks

If most of the problem and issues of 2010 have been solved, there is still some improvement left to be done.

A reference change as a function of time is needed both for tune and orbit feedback (useful for tune change in squeeze, change of crossing angle, separation bump closure during ramp…).

The quality of the measurement should be estimated more precisely before the system decides to use it as input for trim.

2010 run has seen lots of discussions about the impact of the damper on the tune feedback measurement quality and how to reduce its effects. This has to be sorted out.

Fixed displays

There is a large amount of fixed displays permanently sitting on the LHC island screens. Apart from an obvious space problem, it is also more confusing for the team on shift that is given too much information. It would be very useful to define sets of fixed displays by machine mode, that the console manager could show and hide according to the LHC beam mode.

Injection interlocks

For the injection process, many interlocks systems are involved:

In SPS: the software interlocks system, the ring BIC, the extraction BIC and the BQM

In LHC: software interlocks system, ring BIC and Injection BIC with the experiments vetoes.

A simple fixed display with a status o all these involved interlocks would help OP team to diagnose faster the origin of the problem when beam has not been injected.

Software release

It never hurts to repeat that the releases of the operational software have to be well tested (proper test environment should be available). Also, the backward compatibility with previous version should always be checked to avoid unwanted side effects. Of course, the Friday evening releases have to be avoided, and the important changes coming with the release have to be communicated to the relevant persons.

Documentation

For most of the applications available in the control room, there is no associated documentation, or not in an easily accessible place. Although the OP team is familiar with “normal” use of most applications, they often need to ask experts for help for more advanced functions. This could be avoided if appropriate user manuals existed. They would also help for the training of new operators or EICs who have trouble finding sources of information.

CONCLUSION

A long list of requested improvement for many systems is presented in this document. Some of them are very important to reduce the turn around, like the injection software improvement. Some issues also need to be solved to minimize the downtime, like the TCDQ problem and the LSA settings management issues.

Efficiency could also be improved with an optimized nominal sequence. And the risk of error and mistake can be minimized thanks to the state machine and settings checks.

Some minor requests have also been expressed because they can help to diagnose problems more efficiently before calling the expert (RF interlock, diamond improvement), help to detect problems faster (alarms) and improve the ergonomics (dynamic fixed displays).

This represents a big amount of work for developers of controls, OP and equipment group. It will request also a dedicated testing time before LHC is back in operation.

ACKNOWLEDGMENTS

The author would like to thanks the following persons for their help in the preparation of this paper: the whole LHC operation team for the collection of issues and new requirements, Vito Baggiolini , Grzegorz Kruk, Etienne Carlier , Mike Lamont and Stefano Redaelli for their explanations on specific problems.

REFERENCES

[1] S.Redaelli "How to improve the turnaround", these proceedings.

[2] Mike Lamont, Reyes Alemany "LHC sequencer – operational functionality, interface and requirements" EDMS doc LHC-CQ-ES-0001