THE LHC SEQUENCER

Reyes Alemany-Fernandez, Vito Baggiolini, Roman Gorbonosov, Denis Khasbulatov, Mike Lamont, Pascal Le Roux, Chris Roderick, CERN, Geneva, Switzerland

Abstract

The Large Hadron Collider (LHC) at CERN is a highly complex system made of many different sub-systems whose operation implies the execution of many tasks with stringent constraints on the order and duration of the execution. To be able to operate such a system in the most efficient and reliable way, the operators in the CERN control room use a high level control system: the LHC Sequencer. The LHC Sequencer system is composed of several components, including an Oracle database where operational sequences are configured, a core server that orchestrates the execution of the sequences, and two graphical user interfaces: one for sequence edition, and another for sequence execution. This paper describes the architecture of the LHC Sequencer system, and how the sequences are prepared and used for LHC operation.

THE LHC SEQUENCER ARCHITECTURE

The LHC Sequencer Architecture is made of two core components: the Sequencer Executor and the Database. Two Graphical User Interfaces (GUI), one to interface the executor and another one to interface the database, provide the operators in the CERN Control Centre (CCC) with the required control on running sequences and on the creation or modification of sequences, respectively.

The Sequencer Executor part has been extensively presented in [1]. Here a small summary of the main components will be recalled, but emphasis will be given to the database part and the operational sequences used in the daily life operation.

Sequencer Server and Sequencer Client

The architecture and technology used by the Sequencer follows the CERN accelerator controls software standards: a 3-tier architecture implemented in Java using the Spring Framework [2]. This architecture is shown in Figure 1. The sequencer middle-tier server (running Linux) contains the core functionality, mainly, sequence execution, temporary sequence storage and interface via Java libraries to the controls of the different LHC sub-systems, i.e. the LHC Software Architecture (LSA) [3] and the accelerator controls Common Middleware (CMW) [4].

The client tier consists of an execution GUI, shown in Figure 1, from which the execution of the sequences is controlled by the LHC operators in the Linux or Windows consoles at the CCC. Many GUIs may connect to the same middle-tier server.

Database Persistent Storage of Sequences

All sequences are persistently stored in the Oracle database. The database schema consists of a series of tables that map the sequences representation in the following way:

- Sequence table: stores the sequence name, the display name as will appear in the GUIs, brief description, date of creation, name of the person that created the sequence and the sequence category. Each sequence is uniquely identified by a primary key (PK); the name is constraint to be unique since the sequencer server retrieves sequences by name, not by PK.

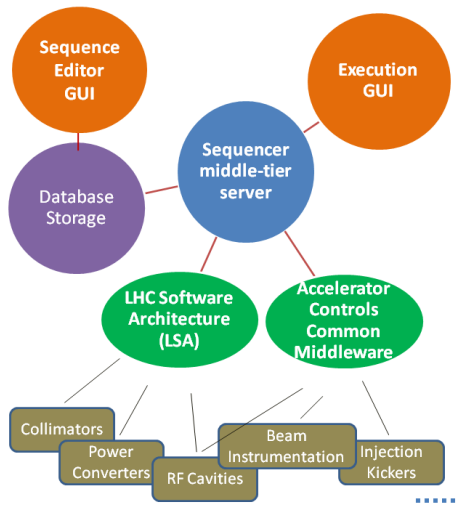

Figure 1: LHC Sequencer Architecture.

- Subsequence table: stores the same information as the sequence table. Each subsequence is uniquely identified by a primary key, and, for the same reason as above, the name is unique.
- Seq-subsequence table: each sequence is made of one or several subsequences. This table contains the list of subsequences that belongs to each sequence. A subsequence can be used by different sequences
- Sequence_components table: each subsequence is made of a number of components which can be of two types: atomic tasks or subsequences. Each atomic task component can be described by a number of parameters which corresponds to columns in the same table. Those parameters are: the name of the hardware group to be addressed by the task (for example a name referring to all the power converters in LHC); tasks can act on a group of hardware or on a single device, therefore another parameter is the device name; the name of the LHC cycle during which the task will have to act, e.g. LHC.USER.INJECTION, LHC.USER. RAMP. If the component is a task, there are a set of separate tables that describe those tasks, uniquely identified by a

primary key which is referred to within the Sequence_components table. Each task component can be further configured to have certain behaviour in case the task fails the execution. The behaviour can be the execution of another task, or another subsequence, or a complete sequence, or simply, stop or continue. Four columns in the components table are reserved to define the on_error behaviour. Once the on_error behaviour is executed by the server, the main sequence can continue execution or stop. This is configured in the same components table via another column. Finally, components are assigned an execution order, an integer number that the server uses to establish the execution order of all the tasks inside the (sub)sequence. If two or more components of type task have the same order, they are executed in parallel at the server level, and only when all the parallel tasks are finished (either successfully or with errors) the server returns the control to the GUI for that particular (sub)sequence. The tasks or subsequences have a default action that can be run, skip or break. All components configured to run are executed (following the established order) when the (sub)sequence is executed; if a component is configured to be skipped, then it will not be executed; if a component is configured as a break component, the execution of the (sub)sequence will be stopped at this component.

- Category table: each (sub)sequence is categorized according to the following criteria:
  - Physics: final version of a (sub)sequence, which has been debugged and it is ready to be used in routine operation.
  - Development: (sub)sequences under construction and debugging, not to be used during routine operation until they are validated and become operational, i.e. of physics category.
  - Machine development: (sub)sequences dedicated to machine development periods.
  - Equipment specific: (sub)sequences to test specific accelerator equipment for commissioning.
- Task type table: within this table, general task configuration parameters can be specified, i.e. parameters that precise if the task will be executed using LSA methods or it is a CMW type task.
- Task instance and task instance parameters tables: once a task type is defined, instances of it can be further created. Each instance can be parameterized to act on a particular property of the task type. For example, a task type frequently used in LHC sets the beam mode. The task type is called SET BEAM MODE and it is a LSA task. Several instances of it exist in the database, each sets the mode to a given value, e.g. SET BEAM MODE TO INJECTION PROBE BEAM, SET BEAM MODE TO RAMP, etc. Within the task instance parameters table the different modes are specified which are columns of the table called, parameter_name and parameter value. Parameter_name in this example is called mode, and the parameter_value is the mode name: INJECTION PROBE BEAM, RAMP, etc.

One of the most important advantages of using the database as persistent storage platform is that the sequences can be edited, modified and new sequences, subsequences and task instances can be created without the need of releasing any software component. The database provides with a very dynamic user interface.

Sequence Editor and Sequence Executor

Every of these tables and columns is filled using the Sequences Editor, a Java program that access directly the database via SQL statements. The editor is protected with Role Based Access [5] in order to restrict the edition of sequences to authorized people only. A picture of the Sequence Editor is shown in Figure 2. On the left hand side panel the user can select the sequence to be modified. There is a filtering panel in order to show only the sequences of a given category. From this panel access to task type and task instances tables are possible, as well as the possibility to create a new sequence. On the right hand side panel different Java tables map the tables in the database and different buttons allow adding the required information. The Java tables are editable.

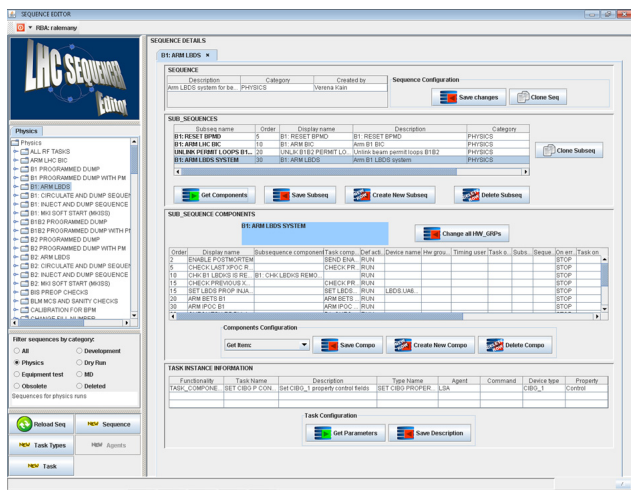

Figure 2: LHC Sequence Editor GUI.

Several of all those columns presented before are freign keys to columns within the LSA database tables. For example, the hardware group column of the table sequence_components, is filled with the content of the hardware_group column of a particular LSA table containing this information. In this way the user cannot enter any name which could not be recognized by the system, but the Editor shows, in the form of a combobox, only the allowed hardware_groups.

Once a (sub)sequence is created or modified by the user, the sequencer server retrieves it from the database and it is converted into a Java source file. The right compilation of the source code is a way of ensuring, to a great extent, the coherence of the sequence.

The Sequencer Executor is a graphical user interface (GUI) developed using the Standard Widget Toolkit[6]. Figure 3 shows a picture of the Executor.

The operator can search for a given sequence, or using the Quick Launch Panel (Figure 4) can search for the most used sequences which are organized by functionality, e.g. Specific Equipment sequences, Experiment sequences, etc. Once a (sub)sequence is selected, the GUI shows the corresponding tree structure of the sequence where the component names displayed are the ones defined by the user as displayed names in the database tables. Clicking on a particular task, the parameters of the tasks, as read from the database, are shown in the tab called "Details". The GUI allows for drag and drop of subsequences.

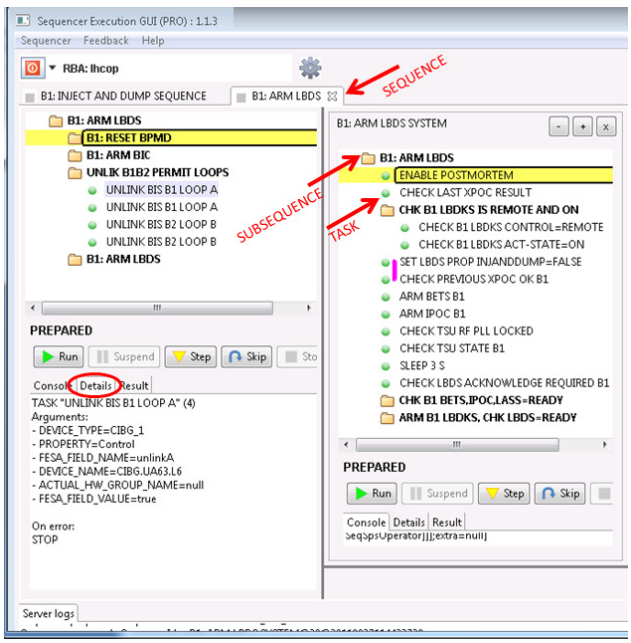

Figure 3: LHC Sequence Executor GUI.

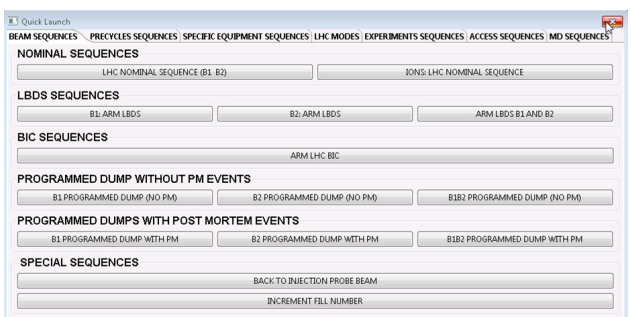

Figure 4: Quick Launch Panel.

In Figure 4 the subsequence B1: ARM LBDS has been dragged and dropped on the right and can be executed as an independent sequence. The vertical magenta bar in the picture indicates that the two tasks called SET LBDS PROP INJANDDUMP=FALSE and CHECK PREVIOUS XPOC OK B1 will be executed in parallel since they have the same execution order in the database. The default action configured in the database (run, skip or break) can be modified online within the executor GUI, but it is not propagated to the database. Modifications of the database configuration can only be done with the Editor. The subsequences can be (un)collapsed to show more or less details of the tree.

USE OF SEQUENCES FOR LHC OPERATION

The LHC operation in routine mode, like luminosity production, certain machine development periods, machine optimization studies, relies fully in the execution of sequences; nothing is done "manually". In the following we explain two cases rather representative of the LHC operation: Inject and dump sequence and LHC nominal sequence.

Inject and Dump Sequence

The distinctiveness of this sequence is that uses a particular functionality of the executor server, the possibility of coming back to a given task after completion of the execution and re-starting again the execution in an automatic way. This functionality is achieved via two tasks, one labels the position within the sequence from where the execution will have to be repeated, the other task instructs the executor to go to the label task. This functionality is very important in this sequence because it allows the arming of the beam dump system and the beam interlock system, injection into LHC and dump in a continuous way when performing injection steering or injection studies.

LHC Nominal Sequence

The LHC nominal sequence contains more than 1000 tasks and is organized in a set of subsequences that maps the LHC cycle: preparation for injection, injection, ramp, squeeze, collisions, and after more than 10 hours of stable beams, programmed dump and ramp down, as depicted in Figure 5. The LHC nominal sequence addresses all the LHC equipment at different steps in the cycle. It is not run in one go as the case above, but every subsequence is run separately, or step by step, since in between subsequences, beam measurements and corrections have to be performed which are not included in the sequence.

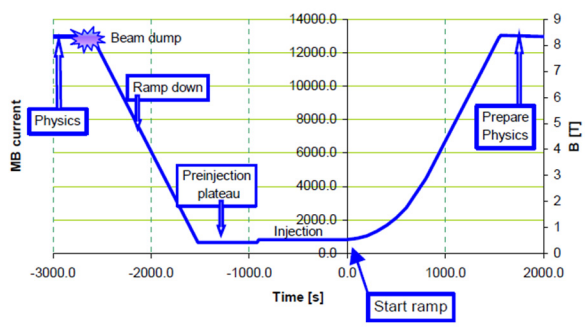

Figure 5: LHC nominal cycle (for 7 TeV flat top). All the different steps in the cycle are driven by the LHC Sequencer.

REFERENCES

[1] V. Baggiolini, R. Alemany Fernandez, R. Gorbonosov, D. Khasbulatov, M. Lamont, "A Sequencer for the LHC era", ICALEPCS'09, Kobe, Japan, 2009, Conference Proceedings.
[2] http://www.springframework.org
[3] G. Kruk, S. Deghaye, M. Lamont, M. Misiowiec, W. Sliwinski, "LHC Software Architecture [LSA] – evolution toward LHC beam commissioning", ICALEPCS'07, Knoxville, Tennessee, 2007, Conference Proceedings.
[4] http://proj-cmw.web.cern.ch/proj-cmw/documents.htm
[5] S. Gysin, A.D. Petrov, P. Charrue, W. Gajewski, V. Kain, K. Kostro, G. Kruk, S. Page, M. Peryt, "Role-based Access Control for the Accelerator Control", ICALEPCS'07, Knoxville, Tennessee, 2007, Conference Proceedings.
[6] http://www.eclipse.org/swt