The TriggerTool Graphical User Interface to the ATLAS Trigger Configuration Database

P. Bell, D. Berge, S. Brunet, G. Fischer, M. Goebel, J. Haller, S. Head, A. Höcker,
T. Kohno, A. Martyniuk, M. Nožička, M. Owen, R. Spiwoks,
J. Stelzer, T. Wengler, W. Wiedenmann

Abstract—A system has been designed and implemented to configure all three levels of the ATLAS trigger system from a centrally provided relational database, in which an archive of all trigger configurations used in data taking is also maintained. The user interaction with this database is via a Java-based graphical user interface known as the TriggerTool. We describe here how the TriggerTool has been designed to fulfill several different roles for users of varying expertise, from being a browser of the database to a tool for creating and modifying configurations.

I. INTRODUCTION

THE three level trigger system of ATLAS (A Toroidal LHC Apparatus [1]) at CERN's LHC is an essential component of the detector, rejecting 99.9995% of the events from the proton-proton collisions while retaining those which may contain interesting physics. The trigger [2] is a complex and highly configurable system. During data taking its complete configuration must be deployed quickly and reliably to all of its hardware and software components. Subsequently, in order to understand, and reproduce as necessary, the trigger behavior when analysing the data, precise knowledge of the trigger configuration used in the collection of the data is required. The ATLAS trigger configuration system has been designed to satisfy these requirements, using a relational database (the TriggerDB) to store the trigger configuration data in a reliable and easily accessible way.

The TriggerTool graphical user interface to the TriggerDB has been developed to fulfill several different roles. It provides not only a means of browsing the TriggerDB but also a way of working with the configuration data, from making small adjustments, for example in response to changes in the online running conditions, to preparing new configurations in a safe and self-consistent way.

II. OVERVIEW OF THE ATLAS TRIGGER SELECTION STRATEGIES

The proton bunches in the LHC will cross at a rate of 40 MHz. The highly selective and efficient three-level trigger

system identifies and accepts interesting physics events, while keeping the overall data rate within the limits of the data flow and storage systems. In order to understand what information is needed to configure the system, we first present a brief overview of the selection strategies used in each of the trigger levels.

A. The Level 1 Trigger

The first level (LVL1) trigger is implemented in custom-made electronics, reducing the event rate to less than 100 kHz. The event selection is based on information from the muon system and the calorimeters. For the Muon Trigger, signals are taken from the muon trigger chambers: resistive plate chambers in the barrel and thin gap chambers in the endcaps. The LVL1 trigger searches for high transverse momentum muons originating from the interaction region, with the trigger decision being based on the multiplicity of muons for each of six programmable transverse momentum (p_T) thresholds. The Calorimeter Trigger aims to identify high transverse energy (E_T) electrons, photons, jets and hadronically decaying tau leptons as well as events with large missing and total transverse energy. Multiplicity information is available for a number of programmable E_T thresholds.

The LVL1 accept decision is made within a latency of 2.5 μ s based on the multiplicity of the muon and calorimeter objects combined in the Central Trigger Processor (CTP). The decision made by the CTP depends on the combination of the different objects received matching any of the required LVL1 trigger items present in the *LVL1 trigger menu*. The trigger menus can be programmed with up to 256 distinct items, with the possibility to reduce the rate associated with a certain item by applying a *prescale* factor.

B. The High Level Trigger

The High Level Trigger (HLT), which consists of the Level 2 (LVL2) and the Event Filter (EF) trigger, is implemented in software and runs on a farm of about 2000 rack-mounted computing nodes. The HLT uses data from all ATLAS-subdetectors. In LVL2, the trigger operates on so-called "Regions of Interest" (RoIs), detector regions defined in terms of pseudorapidity and azimuth found to be of interest by the LVL1 trigger, whereas the EF algorithms, which are executed after complete events are built, have access to the entire event. The LVL2 budget is about 10 ms, the EF budget

P. Bell, S. Head, A. Martyniuk, M. Owen and T. Wengler are with The University of Manchester, Manchester, UK.

D. Berge, A. Höcker R. Spiwoks and T. Kohno are with CERN, Geneva, Switzerland.

S. Brunet, M. Nožička and J. Stelzer are with DESY, Hamburg, Germany.

G. Fischer, J. Haller and M. Goebel are with DESY and The University of Hamburg, Hamburg, Germany.

W. Wiedenmann is with the University of Wisconsin, Madison, USA.

a few seconds, with events accepted by the EF written to mass storage at a rate of about 200 Hz.

The *HLT trigger menu* is composed of a list of physics signatures, each of which is the result of a *chain* of processing through the LVL2 and EF originating from an accepted LVL1 item. Each chain comprises an ordered sequence of trigger signatures or *steps* that are evaluated sequentially. At each step the trigger signature is the logical combination of one or more *trigger elements*. These trigger elements are produced by running a set of *algorithms* on the trigger elements of the proceeding step. As for the LVL1 items, the rates associated with particular chains can be reduced by applying a prescale factor. The algorithmic processing of the trigger elements is controlled by the HLT steering software [3] which runs in the “ATHENA” ATLAS software environment.

III. THE CONFIGURATION OF THE ATLAS TRIGGER

A. Summary of Trigger Configuration Data

Given the selection strategies described in the previous section, the knowledge required to configure the ATLAS trigger system is summarised here. For the LVL1 trigger the data consist of:

- The list of LVL1 *items* in the *LVL1 trigger menu*, against which the combined muon and calorimeter objects are compared by the CTP.
- The logical combination of muon and calorimeter multiplicities in each item.
- The definition of the trigger *thresholds* for which the calorimeter and muon trigger hardware delivers multiplicities to the CTP.
- The *prescales* associated with each item.

In addition, there are several global settings such as LVL1 random trigger rates. Moreover, a representation of the human-readable menu definition for use by the CTP hardware is required. This is produced by the *Trigger Menu Compiler* and also forms part of the configuration data.

For the HLT the configuration data include:

- The list of LVL2 and EF *chains* in the *HLT trigger menu*.
- The decomposition of the chains into *steps* and *trigger elements*.
- The *algorithms* required to process the trigger elements of each step.
- The *prescales* associated with each chain.

In addition, the software *components* (algorithms, tools and services) used by ATHENA to run the HLT generally all have many configurable *parameters*, the settings of which must be specified. Since the performance of a certain algorithm configured with the same settings may vary from one version of the software to the next, the software *release* information must also be recorded if the HLT behaviour is to be reproduced.

B. The TriggerDB

In the ATLAS trigger configuration system the configuration data are stored in a relational database known as the *TriggerDB*. The complete schema for the TriggerDB is reproduced in Fig.1. All tables have an integer primary key, which

generally corresponds to a unique combination of a name and version for the object being stored. The primary key of the top-level SUPER_MASTER_TABLE, which uniquely identifies one LVL1 and one HLT trigger menu, is referred to as the *supermaster key*. Since it is required that during data taking it be possible to change the prescales without changing the menu, the LVL1 and HLT menus specified by a supermaster key can be used in conjunction with multiple different prescale sets. Thus, to fully specify a trigger configuration three keys are required in all: the supermaster key plus the L1 and HLT *prescale set keys*.

The supermaster table defines one LVL1 and one HLT configuration through its foreign-key relationships to the respective MASTER tables. The relational structure of the tables below the master tables store the objects of the LVL1 and HLT menus and link them in a hierarchical (parent-child) order through “many-to-many” link tables. Thus, on the HLT side, we find the TRIGGER_MENU table as the parent of the TRIGGER_CHAIN table, and then below this the tables for the trigger signatures, trigger elements, algorithms and finally the configurable parameters of the algorithms. Within the software framework a certain component may also be the parent of another component (e.g an algorithm with a private tool) hence the component-to-component link table. In addition to the actual trigger algorithms and their tools, certain *infrastructure components* are also required to run the trigger software, these being linked directly from the HLT master table. Since the performance of the HLT software is also dependent on the particular release installed, the supermaster key is also linked to a RELEASE table where this information is recorded. For the LVL1, below the TRIGGER_MENU table are the tables for the trigger items, the thresholds and the threshold definitions. Further link tables from the threshold table define the CTP cabling information and the configuration of the CTP monitoring. The output of the Trigger Menu Compiler is stored in the CTP_SMX and CTP_FILES tables.

The TriggerDB schema has been designed with two underlying concepts in mind: that there should be no unnecessary *duplication* of any records and that a *history* of all configurations must be maintained. The use of the many-to-many link tables is fundamental to this design. For example, the configuration of a chain with a certain name and version corresponds to a unique record in the database (with a unique primary key). The inclusion of that chain in multiple menus is achieved through the creation of multiple links pointing to it, not from the duplication of the chain object itself, which avoids any ambiguity in its definition. A secondary advantage of this approach is that it economises on the total size of the database. In order to preserve the history of the configuration data, records themselves are never deleted but a child may be removed from its parent (a chain from a menu, for example) by breaking the corresponding link. As the sole means of manipulating the data, the TriggerTool user interface has been designed adhering to these concepts, as will be explained in section IV.

C. System Overview

A simplified overview of the full ATLAS trigger configuration system is presented in Fig. 2. A full description can be found in [4]. The TriggerDB lies at the heart of the system. For configuring the online trigger system the database is hosted on a central Oracle server, which is replicated to a read-only account for offline access.

Fig. 2. Simplified overview of the trigger configuration system. The TriggerDB relational database lies at the centre, which the user may interact with via the TriggerTool. The client software extracts the information to configure the LVL1 hardware and HLT software online, or the offline simulation software.

Custom client interfaces deliver the configuration information either to the LVL1 hardware and HLT software of the actual ATLAS trigger or to the offline trigger simulation software. In this way Monte Carlo (MC) simulation or offline reprocessing can gain access to the same configurations used online.

The TriggerTool provides the user interface to the TriggerDB for the different use-cases as illustrated. A description of the TriggerTool forms the remainder of this paper.

IV. THE TRIGGER TOOL GRAPHICAL USER INTERFACE

A. Introduction

The TriggerTool graphical user interface to the TriggerDB is a standalone project written in Java. It was developed originally in the Eclipse and more recently the NetBeans Integrated Development Environment. It consists of approximately 160 classes with contributions from around 10 different authors.

B. The connection screen: user levels and access control

The TriggerTool may be launched in a number of ways. It is part of the installed software in the ATLAS Control Room (ACR) with a desktop icon on the Trigger Desk. A Java web-start is also available, which is linked from the ATLAS “Run Summary” pages. Alternatively, the package can be checked out of the ATLAS CVS repository and built in the usual way.

The operations permitted on the TriggerDB by the TriggerTool depends on the privileges of the user and the database to which they are connecting. When the TriggerTool is launched

the user is presented with a *login window* where they choose from one of three following access levels:

- **User:** In this mode the TriggerTool connects to the read-only offline replica of the online TriggerDB. Correspondingly, all editing features are disabled.
- **Shifter:** This is the mode used by the Trigger Shifters in the ACR. The TriggerTool connects to the online TriggerDB but restricts the range of allowed operations.
- **Expert:** In this mode, restricted to a small number of Trigger Experts, full editing of the online TriggerDB is possible.
- **DBA:** In this Data Base Administrator mode, in addition to full editing capabilities access roles may also be assigned to other users.

Access is controlled by means of an additional table in the database which stores the names of the allowed Shifters and Experts. These two roles with editing capability are valid only inside CERN, since elsewhere the online Oracle database is not visible. From the Java web-start, which may be used outside CERN, only the read-only User mode is available and the connection is made automatically to the offline replica TriggerDB, the login window being bypassed.

In addition to the Oracle online TriggerDB and its replica, the TriggerTool can be used in conjunction with MySQL or SQLite implementations of the database. Users may therefore create their own TriggerDBs based on their technology of choice and configure the TriggerTool to connect to this under the *My Database Connections* option. Users who create their own database have full editing (Expert) rights there.

C. The main panel: finding configuration information

Once the user has connected to the TriggerDB the *main panel* is presented (Fig. 3). This looks similar regardless of the user-level, though certain buttons are ghosted in User and Shifter mode. By default the *summary table* in the top half of the main panel shows the list of all available supermaster tables in the database. The information displayed includes the primary key (supermaster key), the name and version, plus a comment, the origin of the configuration, the creator and the software release with which it is compatible.

The search field allows the user to search for supermaster tables by any of their properties. An advanced search feature allows supermaster tables containing a certain child record matching some criteria to be found. It is also possible to jump to any level in the hierarchy of the TriggerDB schema using the drop down box in the top left, thereby searching for particular records, chains, items, etc, independent of the configurations in which they may be contained.

In the bottom left part of the main panel the hierarchical arrangement of the TriggerDB tables is reflected in a tree-view of the configuration data. The record at the top of the tree corresponds to the object selected in the summary table above, in this case a supermaster table. The user may then navigate through all the child records, in this case allowing the complete LVL1 or HLT menu to be expanded. For the record currently highlighted in the tree, its properties are shown in the *details table* in the bottom right of the main panel.

D. The overview panel: accessing a full configuration

Doubling-clicking on a particular supermaster entry in the summary table causes the TriggerTool to read all the records belonging to that configuration from the TriggerDB into memory. The complete configuration in memory is then displayed in the *overview panel* (Fig. 4). In the case where the TriggerTool is launched from an ATLAS Run Summary page, the overview panel is automatically presented for the configuration which was used in the run being accessed.

Within the overview panel all of the information about the loaded configuration may be accessed. Reflecting the logic of the three trigger levels, the menu is represented in three columns for the LVL1 items, the LVL2 chains and the EF chains. Lines are drawn to represent the seeding of the L2 and EF chains by the preceding L1 items or LVL2 chains, respectively. The graphical representation of every record is initially drawn contracted, with just the name displayed, but can be expanded to reveal the full list of properties and any child records. In the example picture one HLT chain has been expanded to reveal its three signatures, for one of which the single trigger element and the corresponding algorithms are visible. On the far left a unique list of all LVL1 thresholds, independent of their use in the items, can also be hidden/shown using the *Hide/Show Thresholds* button. A search feature is under development to quickly locate the presence of a particular child record in the configuration.

In addition to the composition of the menu, the overview panel allows the LVL1 global parameters such as random rates and deadtime settings to be changed. These values appear in a summary table (not shown in the figure) which can be hidden/shown using the *Hide/Show Tables* button. Finally, for the HLT, the list of all the ATHENA components needed to run the menu and the settings of their configurable parameters can be found in the *HLT setup viewers* from the *LVL2 Setup* and *EF Setup* buttons.

E. The prescales viewer and editor

The *prescales viewer* (Fig. 5) can be opened from the *Prescales* button on the TriggerTool main panel when a supermaster key is selected in the summary table. The viewer contains two tabs, for the LVL1 and HLT prescale sets. In one tab, the user is presented with the list of prescale sets compatible with the menu associated with the selected supermaster table. For a chosen set the individual prescales applied to the L1 items or HLT chains are then shown in the table on the right. In the case of the HLT, the *pass through* settings are also shown.

If the user is logged on as a Shifter or Expert the prescales viewer becomes an editor of the prescale sets. In this case the prescale values in the table can be adjusted individually for any item or chain. When the user clicks *Save* the TriggerTool creates a new prescale set containing the requested values, and a new prescale set key is generated. Note that a new set is created, an existing set is never modified, in accordance with the need to preserve the history of the configuration data. However, since we also require that no record be duplicated, should the modified set correspond to a set which already

exists the user is simply notified of the unique key of that set. The user may at any time add or edit the comment associated with a particular prescale set.

During data-taking it is foreseen that the editing of the prescales will be the only Shifter-level operation allowed on the TriggerDB by the TriggerTool. This restriction will ensure that all supermaster keys in the database retain their validity. However, the prescale editor alone is a powerful tool for tuning the trigger menus. In addition to adjusting individual rates, whole sections of the LVL1 or HLT menu may be disabled by setting the corresponding prescales to a negative value, which can be done with the *In/Out* check-boxes. The negative values are interpreted by the client software as masks on the corresponding LVL1 items or HLT chains. The TriggerTool knows that should a LVL2 chain be disabled it should automatically disable any seeded chains in the EF, or else the resulting configuration would not be valid.

F. The overview panel as a configuration editor

For a user logged in with Expert privileges, the overview panel becomes a full editor of the complete trigger configuration held in memory. For any object, its properties may be clicked on and changed and child records, if applicable, added or removed. In this way, HLT chains may be added to the HLT menu, trigger elements removed from a signature, etc. In Expert mode the HLT setup viewers also allow the configurable properties of all the software components to be adjusted, though not at this time the addition or removal of components.

Any editing performed in the overview panel takes effect on the in-memory configuration and nothing is written to the TriggerDB until the user clicks *Save*. Only at this point, when it knows which aspects of the configuration have been changed, does the TriggerTool create a new configuration, indexed by a new supermaster key, in the TriggerDB. Adhering to the principle that no objects should be duplicated, any unaltered objects will be exactly those which appear in original configuration but now with appropriate new links made to the new supermaster table. Links are also created for the objects which are new to the particular configuration but exist already elsewhere in the TriggerDB; only for genuinely new objects is any new configuration data written to the database. For objects which have been removed from the configuration, respecting the need to maintain the history of configuration data, the corresponding links are not made to the new supermaster table. Just as for the saving of the prescale sets, if the new configuration is identical to one which already exists the save routine returns that supermaster key to the user: this is the natural consequence of not allowed any duplicate data and provides one simple test that the TriggerTool is performing correctly.

When a new supermaster table is created in the database through the editing and saving of an existing configuration, the prescale sets of the edited configuration are automatically made compatible with the new one. To help keep track of the new configurations being created, the *origin* property of the new supermaster table is always set to the supermaster

Fig. 3. The main panel of the TriggerTool shown in User mode. All available supermaster tables (configurations) in the TriggerDB are shown in the summary table in the top half of the screen, with the most recently created appearing first. In the bottom half, the tree view has been expanded to show the details of one particular chain of the HLT menu, revealing its steps, trigger elements and associated algorithms. Details of the selected record, the HLT chain, are shown in the table on the right.

Fig. 4. The overview panel of the TriggerTool shown in User mode. The display as interpreted from left to right reflects the logic of the three trigger levels, with the LVL1 items (left column) seeding the LVL2 chains (middle column). On the far left can be seen the unique list of all LVL1 thresholds in the menu.

Fig. 5. The prescales viewer. Under the HLT prescales tab the list of available prescale sets for use with the selected supermaster table (configuration) is shown. Choosing one of these from the list, the settings for each chain are shown in the table. Similar information for LVL1 items is available under the L1 prescales tab.

key of the edited configuration. To fully understand the exact differences between the configuration associated with one supermaster key and that of another, a comprehensive and fast *Diff* feature is provided.

G. The TriggerTool in the offline world

In addition to working with the online TriggerDB, the TriggerTool can be set up to work with any private DB which has the TriggerDB schema installed. Indeed, the initial testing of the trigger configurations does not take place from the central database, which is reserved for completed configurations which have been approved and validated for data-taking or MC production. A *DB Copy* feature is therefore implemented to allow the copying of configurations from one database to another which allows, for example, validated configurations to be copied to the online database by the Experts.

The DB Copy method may also be used by any User to copy a configuration from the online DB to their private database, where they will then be able to edit it. This allows the user, who may be performing physics analysis or perhaps developing the trigger software themselves, to study the effects of their own changes on a particular configuration. In this way, the TriggerTool has the potential to go beyond being a means of operating the Trigger in the ACR to being a tool for wider use in the offline world.

At the time of writing, an additional central Oracle database, the MC TriggerDB, has been established for storing pure offline configurations for use in MC production. These configurations are not intended for use online so do not belong in the online Oracle database. In the ATLAS computing model, MC production takes place on the grid at the Tier 2 sites. This

requires that the MC TriggerDB be replicated to the remote sites, where the Oracle DB hosted at CERN is not visible. This replication is achieved using an *Exact DB Copy* mechanism in the TriggerTool. This mechanism allows a subset of keys from the Oracle MC TriggerDB to be selected for replication to an sqlite file which is then shipped to the Tier 2 sites. The copy is *exact* in that the primary keys of all tables (including that of the supermaster table) are preserved. Thus, supermaster key *S* refers to the same configuration in both the central and replicated databases, even if not all the keys with number below *S* were replicated.

H. Configuration preparation

The LVL1 and HLT physics menus are initially constructed using a set of python tools, which can create an XML representation of the desired menu. The XML files are then uploaded to the TriggerDB using the *XML Upload* function of the TriggerTool. (A download function is also implemented.) For the LVL1, additional steps are required in order to obtain the full set of configuration data needed to realise the required menu in the trigger hardware: the output of the Trigger Menu Compiler must be linked to the menu in the database and the CTP cabling information created. These steps are performed in the TriggerTool in Expert mode.

Once the initial configuration is uploaded, the TriggerTool offers a safe and convenient way to make any further changes to the menu as stored in the TriggerDB. As described in the previous section, the overview panel allows any aspect of the configuration to be edited. The editing performed there benefits from several types of *consistency checking* by the TriggerTool. The first checks are actually performed during the

XML upload, when the TriggerTool checks the basic validity of the LVL1 and HLT menus independently. For example, multiple trigger elements of the same name are not allowed in the same HLT menu, as such an ambiguity could not be understood by the HLT steering software. Once the two menus are uploaded and a supermaster key created, the overview panel shows any inconsistencies between the two levels: any HLT chains requiring LVL1 seeds that are not present in the LVL1 menu are collected together under a list of invalid chains. Finally, a *Consistency Check* button on the overview panel can be used to call a full range of checks based on the TriggerTool's knowledge of the LVL1 hardware and HLT software requirements.

Since the TriggerTool has been designed to offer an easy and safe way of working with the trigger menus, the ambition for the future is to investigate migrating the menu building work entirely to the TriggerDB and retire the custom python framework.

V. SUMMARY

The ATLAS trigger is essential for reducing the data rate from the LHC by identifying events of potential interest from the background. The data needed for the configuration of the trigger must be both reliably deployed to the LVL1 hardware and HLT software for data taking and made easily accessible for faithful reproduction in data analysis or simulation. The trigger configuration system has been designed to address these needs, through the use of a relational database for preparing, storing and archiving the configuration data.

The TriggerTool provides the sole means of user interaction with the TriggerDB. It is a tool for any User to browse the history of the configuration data; for Shifters to edit the trigger configuration data in the ACR in a controlled way; and for Expert to prepare and edit trigger full trigger menus. In the offline world it also allows any user with a private database to copy, study and tune the configurations used online. The design of the TriggerTool has been closely coupled to that of the database schema, such that operations on the TriggerDB never introduce any duplication of the existing records and preserve the history of the existing data. The TriggerTool has an essential role in the operation of the ATLAS trigger, as already demonstrated in the commissioning phase.

REFERENCES

- [1] *ATLAS Technical Proposal*, CERN/LHCC/94-43, 1994.
- [2] *ATLAS HLT DAQ and Controls*, CERN/LHCC/2003-022, 2003.
- [3] N. Berger *et. al.*, *The High-Level-Trigger Steering of the ATLAS Experiment*, IEEE Trans. Nucl. Sci., vol. 55, no. 1, pp165-171, Feb. 2008.
- [4] A. dos Anjos *et. al.*, *The Configuration System of the ATLAS Trigger*, IEEE Trans. Nucl. Sci., vol. 55, no. 1, pp392-398, Feb. 2008.