UNIVERSITY
OF SUSSEX

Data acquisition software development and physics studies for a future e^+e^- linear collider

Tom Coates

Submitted for the degree of Doctor of Philosophy

University of Sussex

June 2019

Declaration

I hereby declare that this thesis has not been and will not be submitted in whole or in part to another University for the award of any other degree.

Signature:

Tom Coates

UNIVERSITY OF SUSSEX

Tom Coates, Doctor of Philosophy

Data acquisition software development and physics studies for a future e^+e^- linear collider

Summary

[Summary text] [Max. 300 words for most subjects]

Acknowledgements

Contents

Chapter 1

Introduction

> Teaching man his relatively small sphere in creation, it also encourages him by its lessons of the unity of Nature.
>
> Annie Jump Cannon

[...]

1.1 The Standard Model

[...]

1.2 The Higgs Boson

[...]

1.3 CP violation in the Higgs sector

[...]

1.4 Data acquisition software and testbeams

[...]

Chapter 2

Future Linear Colliders

Progress is not a straight line.

An Wang

In the post-LHC era, the major unanswered questions in particle physics centre around the Higgs boson and its properties, the identification of additional sources of CP-violation that can account for the universe's abundance of matter and paucity of antimatter, and the discovery of physics beyond the Standard Model. There are many investigations into each of these fields that utilise the Large Hadron Collider, or will leverage the upgrades for the high-luminosity LHC (HL-LHC). However, now that the Higgs boson has been identified successfully, one of the most [x] avenues for further research is the construction and operation of a lepton collider with sufficient centre-of-mass energy to produce Higgs bosons. [...]

These are the primary motivations for the construction of future colliders, especially lepton colliders, to succeed the Large Hadron Collider. The two main candidates are the International Linear Collider (ILC) and the Compact Linear Collider (CLIC). Since both are linear eletron-positron colliders, they share many features, design considerations, and challenges,

2.1 Introduction

[...]

2.2 The physics case for a lepton collider

[...]

Measurements of particles and their properties will be made significantly easier and thus more precise in the cleaner final state environment of a lepton collider, allowing for precision measurements of the Standard Model, putting better limits on the existence of new physics. It will also allow high-precision measurements of the properties of the Higgs boson, allowing linear colliders to determine their properties very precisely. Multiple BSM models rely upon properties of the Higgs being different from those predicted by the Standard Model, and

2.3 The International Linear Collider

The International Linear Collider (ILC) is a proposed high-luminosity linear electron-positron collider, designed to have an initial energy of between 200-500 GeV, upgradable to 1 TeV at a later date. The ILC and its detectors are designed with the intention of becoming a "Higgs factory" – producing large numbers of Higgs bosons to allow more detailed study of these particles.

There were a number of proposed sites for the ILC, including CERN in Geneva, DESY in Hamburg, and JINR near Moscow. Due to the 2008 economic crisis, the United States and United Kingdom severely cut funding for linear collider projects, and this resulted in Japan stepping up to offer to host the collider in the Kitakami Highlands region of the Iwate prefecture. This is partially due to a commitment the Japanese government made in [year] to cover half of the cost of construction, commissioning and operation if it was hosted within Japan.

However as of writing, a report from the Science Council of Japan (a representative organisation of the Japanese science community) released in 2019 expressed that they have not reached a consensus as to whether to support hosting the ILC in Japan. Some of the reasons cited were concerns over international cost-sharing in the long-term, as well as to whether the expected scientific outcomes would justify the unprecedented human resource requirements and infrastructure necessary to make the ILC a reality [1].

The final decision to host the ILC will be made by the Japanese government's Ministry of Education, Culture, Sports, Science and Technology (MEXT), [...]

2.3.1 The ILD and SiD detectors

[...]

One of the unique features of the ILC is the push-pull detector system. This is a moving platform in the chamber housing the interaction point, upon which two detectors can be

mounted. The platform can be moved to change which detector is in the beamline, allowing a linear collider to function with multiple detectors. Switching detectors is expected to take [some] hours. This allows the two detectors to specialise for different physics studies and goals, much like the experiments at the Large Hadron Collider at CERN, which is normally not possible with linear colliders. [?] [...]

The International Large Detector (ILD)

[...]

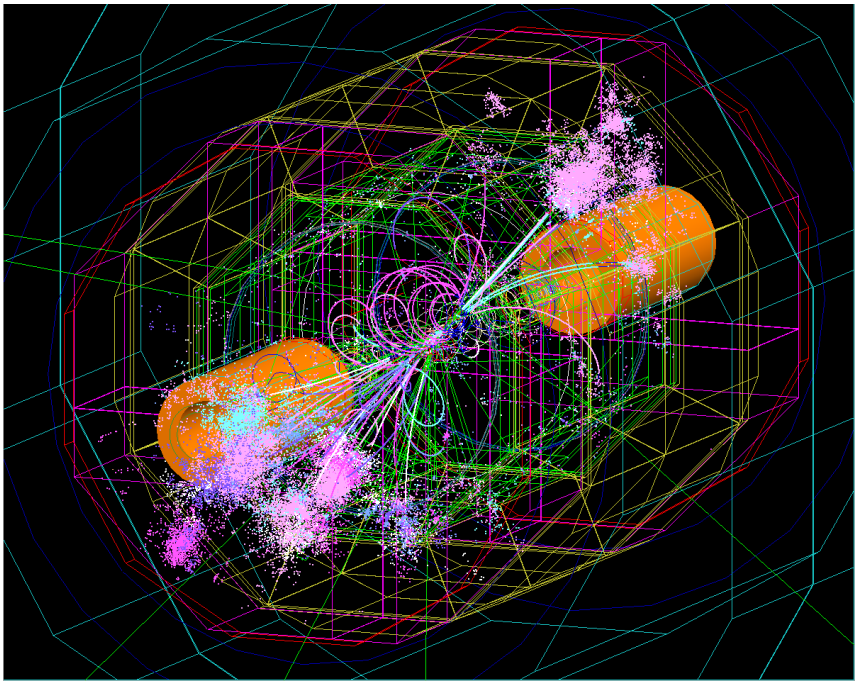

Figure 2.1: Visualisation of a simulated tth event in the ILD. Charged particles can be easily identified by their curved, coiled or spiral paths, and the jets are clearly visible as the light pink and purple areas near the beampipes on either side.

The Silicon Detector (SiD)

[...]

2.4 The Compact Linear Collider

[...]

[...] CLIC would be built beneath the existing LHC ring at CERN, stretching across the French-Swiss border and running parallel to the feet of the Jura mountain range. [...]

As of writing, the CLIC project has been submitted as input for the European Particle Physics Strategy Update, which will decide which projects the CERN collaboration chooses to pursue from 2020 onwards. [...]

CLIC's initial centre-of-mass energy will be 380 GeV, with successive upgrades increasing it to 1.5 TeV and 3 TeV.

Chapter 3

Data acquisition software

Before software can be reusable
it first has to be usable.

Ralph Johnson

Data acquisition is a critical component of all particle physics experiments across all stages of technological readiness, from the very beginning of hardware testing in tabletop experiments to full-scale international experiments like the Large Hadron Collider.

In the modern era of particle physics, the interplay of hardware and software at minscule timescales drives everything, and almost all results are highly dependent upon the speed and efficiency of the electronics and computer systems that extract data from the detectors. A massive quantity of work goes into creating, testing and optimising the systems that will acquire, process, sort and transport data before it is ever seen by the physicist operating the experiment.

Of particular interest in this thesis is the data acquisition software during the development phase, where individual detector subcomponents are undergoing prototyping and testing. These development and iteration cycles are tied closely to testbeam facillities such as the Super Proton Synchrotron (SPS) at CERN and the DESY II synchrotron at DESY. At this point in the development cycle, the detectors are beginning to take shape and this is where data acquisition (or DAQ) becomes an important consideration.

In addition to this, the data acquisition solutions used during the testbeam phase of detector development is likely to inform the final data acquisition solution, either directly by evolving into the final software, or indirectly by identifying and evaluating the particular features or challenges of the subdetector components that the software must take into account or accommodate.

During this stage, each individual detector component – such as a vertex tracker or

hadronic calorimeter – will be developed by small teams, and the natural tendency is for each of these groups to set their own standards and develop their own tools, prioritising the features that are important to their specific case. However, in the past this approach has generated a variety of *ad hoc* solutions for testbeam software, many of which cannot be applied outside of their original scope. The also results in wasted effort and time, as different teams implement the same solutions anew for each subdetector.

One of the aims of the AIDA-2020 project is to improve this situation by developing generic and reusable software tools for testbeams and particle physics experiments.

The AIDA-2020 project

The AIDA-2020 project is an EU-funded research programme for developing infrastructure and technologies for particle physics detector development and testing, comprising 24 member countries and lead by CERN.

The overarching goal of the project is to develop common infrastructures and tools for physics testbeams, and software is one such important tool. By creating a suite of tools that are designed with a variety of uses in mind, the amount of effort and development time necessary to plan and implement data acquisition and monitoring setups can be significantly reduced or eliminated, speeding up the planning and deployment of physics testbeams. This allows more science to be done faster. The two tools within AIDA-2020 that facillitate this are EUDAQ and DQM4hep, discussed in more detail below.

3.1 Overview of DQM4hep

Data Quality Monitoring for High-Energy Physics (abbreviated DQM4hep) is an online monitoring and data quality monitoring framework developed for physics testbeams for high-energy and particle physics. It is designed to be able to fulfil the requirements of monitoring for physics testbeams in a generic way. The framework is written in the C++11 standard and can run on any Linux distribution. The only requirements for installation are a compiler compliant with the C++11 standard, cmake 3.4 or higher, and ROOT 6. All other dependencies are downloaded and compiled automatically during installation.

The two core principles of DQM4hep are genericity and modularity. The core of the system is based on a plugin system that allows shared libraries to be loaded and hook classes for further use [2].

This structure allows for independent components of the framework to be used, not used, or exchanged, by isolating each function of the program into specific and independent

processes. The components that are specific to particular users are written by those users, and the rest of the framework then handles packaging this information in a useful way and networking to transmit it to where it is needed, meaning that the user does not have to worry about the mechanics of data storage, serialisation or transmission.

The experiment-specific components have to be written by the user, but these components use standard C++ code with a few DQM4hep-specific functions to handle their integration into the framework, making them easy to understand for users who already have experience coding in C++. This also means that the framework is capable of working with any data format that can packed into, decoded from, and accessed with normal C++ methods, including those that can be loaded from external libraries. This results in a framework that is able to deal with any kind of data, including user-defined data types, making it more flexible, portable and easily reusable.

3.1.1 Architecture

DQM4HEP is designed with genericness as its core paradigm, using processes and algorithms that are independent of data type. The ability to run multiple instances of each process of the framework is also key to its flexibility, allowing users to, for example, separate sub-detector data from data that has undergone event building, operate in online or offline modes, or distribute the computational load of the analysis over several networked computers.

The generic nature of the framework lies in two core features:

- The Event Data Model abstraction allows the user to define the type and structure of an event and how serialisation should be handled.
- The plugin system allows the inclusion of any user-defined classes via external libraries, such as to select the serialisation process, online analysis, etc.

The plugin system for end-users consists of four different types of plugins: analysis modules, standalone modules, file reader plugins, and file streamer plugins. Each of these will be discussed in-depth in the sections below.

A diagrammatical representation of the overall structure of the framework can be seen in Fig. 3.1.

Analysis modules

Analysis modules recieve events from the data aquisition system, processing the data according to a user-specified procedure to create ROOT TObjects like histograms, graphs,

plots, etc. The analysis module then handles encapsulating these objects as monitor elements, and sending them to the rest of the framework for display and storage.

An analysis module is specific to one use case, and is intended to be written by the user with their data format and processing needs in mind. However, the framework provides both templates and examples for how to write an analysis module.

An example of the structure of the framework utilising an analysis module can be seen in Fig. 3.2.

Standalone modules

Standalone modules are identical in form to analysis modules described above. The distinction is that a standalone module does not operate on data coming from the data aquisition device. One of the intended and most common usages of standalone modules is as a slow control, taking data from monitoring sensors on the device rather than data, to report on the condition of the hardware. Standalone modules could also be used to *generate* data, if needed, acting as a programmed signal generator or random number generator.

An example of the structure of the framework utilising a standalone module can be seen in Fig. 3.3.

File reader plugins

A file reader is a type of plugin that reads a file from the disk and packs it into a data structure necessary for usage within DQM4hep. They are used primarily for offline monitoring or data processing. File readers can be made for any kind of file, provided the user understands the data structure. There are existing examples of file readers for data stored as binary, plain text, LCIO files, and ROOT TTrees.

An example of the structure of the framework utilising a file reader plugin can be seen in Fig. 3.4.

File streamer plugins

A file streamer is a type of plugin that reads data from a stream and packs it into a data structure necessary for usage within DQM4hep. They are for receiving data from a data acquisition device for online monitoring. File streamers can be made for any kind of data stream, provided the user understands the data structure. File streamers are considered the “default” in DQM4hep.

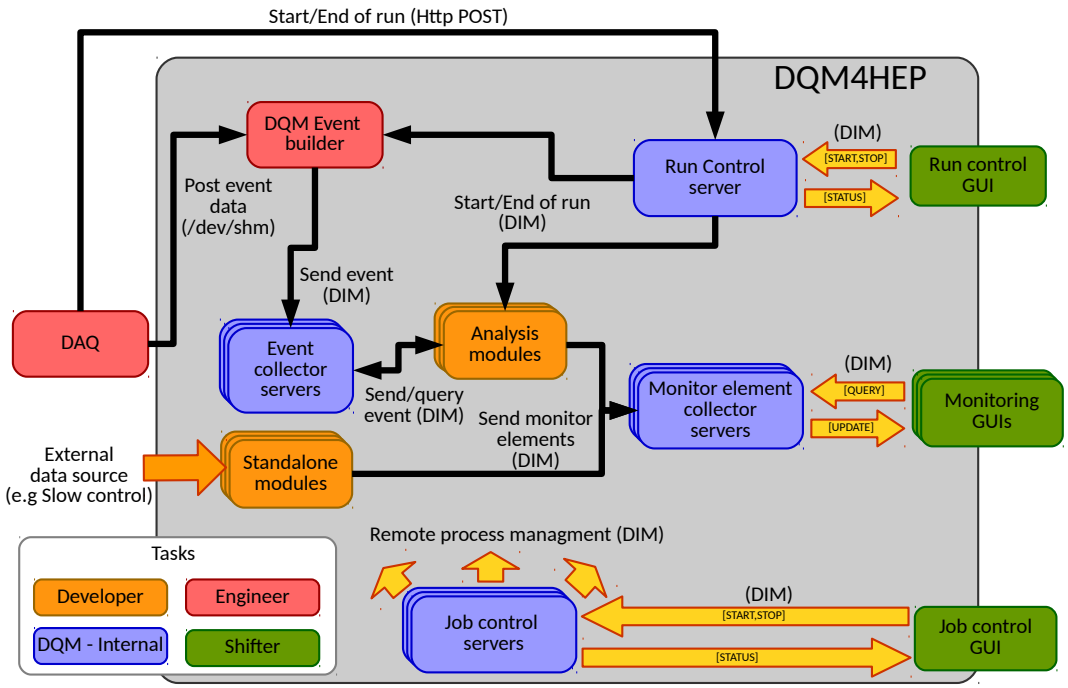

Figure 3.1: The global online architecture of DQM4hep. Each block is colour-coded to show which operator of the testbeam is responsible for the process.

3.1.2 Visualisation and graphical user interface

As of writing, the graphic user interface and visualisation suite is still under active development. Previous versions of DQM4hep have utilised the Qt framework for GUI and visualisation.

The decision to remove the Qt-based GUI from the framework was made for two reasons. Firstly because of integration with ROOT – running DQM4hep with a Qt-based GUI requires an installation of ROOT compiled with the `--enable-Qt` flag enabled, and the majority of ROOT installations in remotely-accsible file systems based at CERN and DESY (which are heavily used for analysis and testbeams) were not compiled this way. Secondly, Qt was an additional dependency that must be installed prior to use, making the software more dependent upon the operating system and environment of the machine, and thus less generic and easy to use.

However, the removal of the Qt-based UI however allows for greater freedom with UI development. The intended goal is to have a browser-based GUI, removing dependency on specific software frameworks, allowing it to function on any device. This will also make it more compact and portable, as the interfaces for run control, networking, and data monitoring and quality display can be simply run in different tabs of the same browser window.

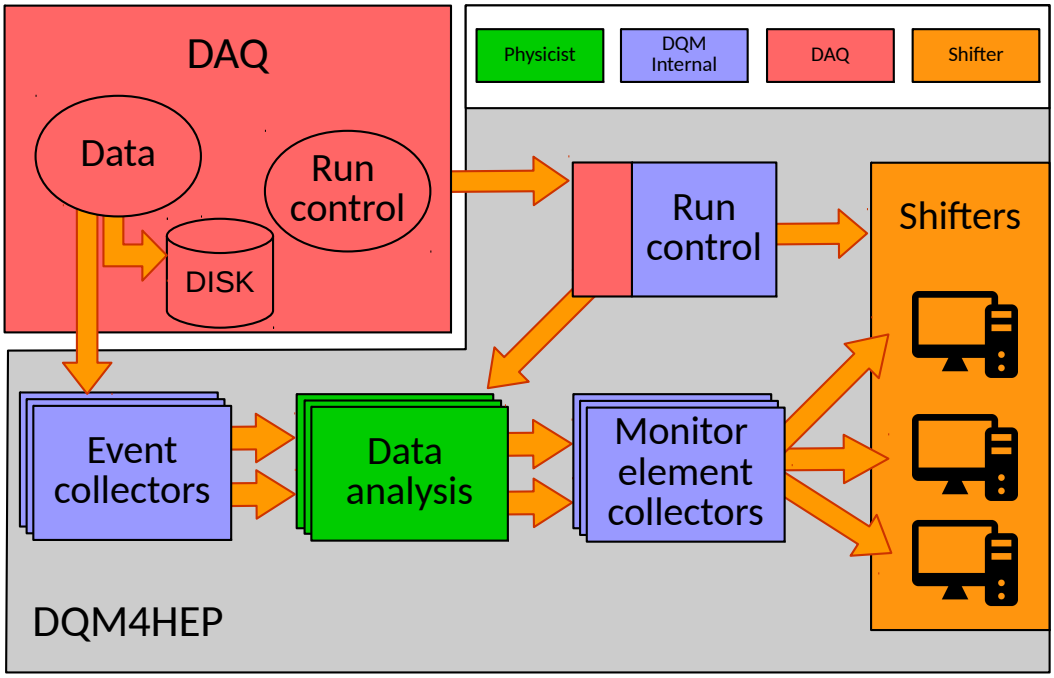

Figure 3.2: The structure of running DQM4hep online using an analysis module.

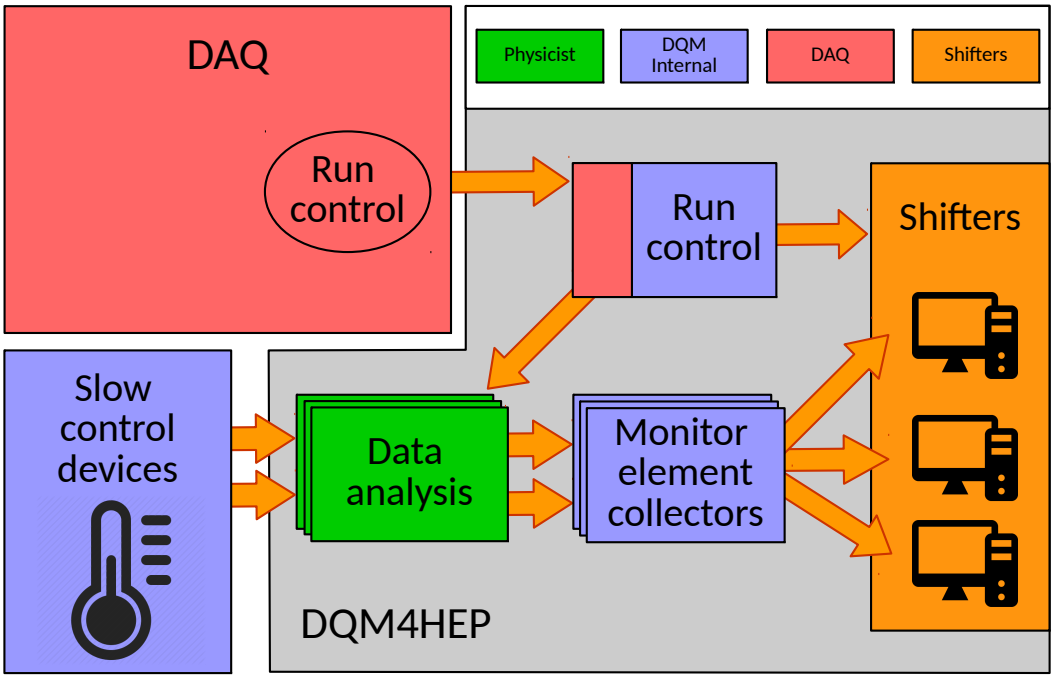

Figure 3.3: The structure of running DQM4hep online using a stand alone module.

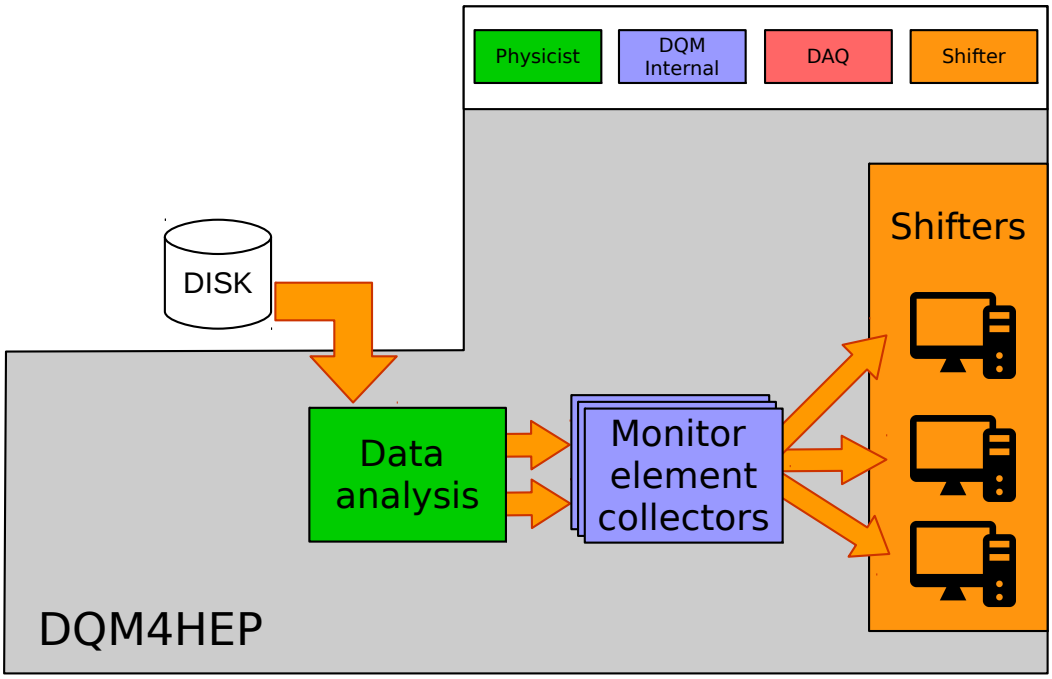

Figure 3.4: The structure of running DQM4hep online using a file reader module.

An example of the older Qt-based monitoring GUI in use can be seen in Fig. 3.5. A mock-up of the web interface can be seen in Fig. 3.6.

3.2 Data quality testing

One of the important areas of DQM4hep that was not yet completed was data quality monitoring, which is an array of tests or programs that assess the data being taken in real time to allow testbeam operators and shifters without detailed knowledge of the hardware, software, or physics to determine whether the device under test is performing as intended, and to quickly identify and address any errors or inconsistencies. Data quality monitoring (DQM) uses a variety of methods for measuring the 'quality' or 'goodness' of data, mainly relying upon statistical or comparative methods.

DQM4hep did not have any infrastructure to support data quality tests, but this was added during refactoring for the next release version. Once this was in place, a variety of data quality tests were developed and implemented, detailed below.

3.2.1 Property within expected test

This is a quality test that takes either a TH1 or TGraph object, and finds some user-defined parameter. The parameter must be one of: mean, mean90, root mean square (RMS), root

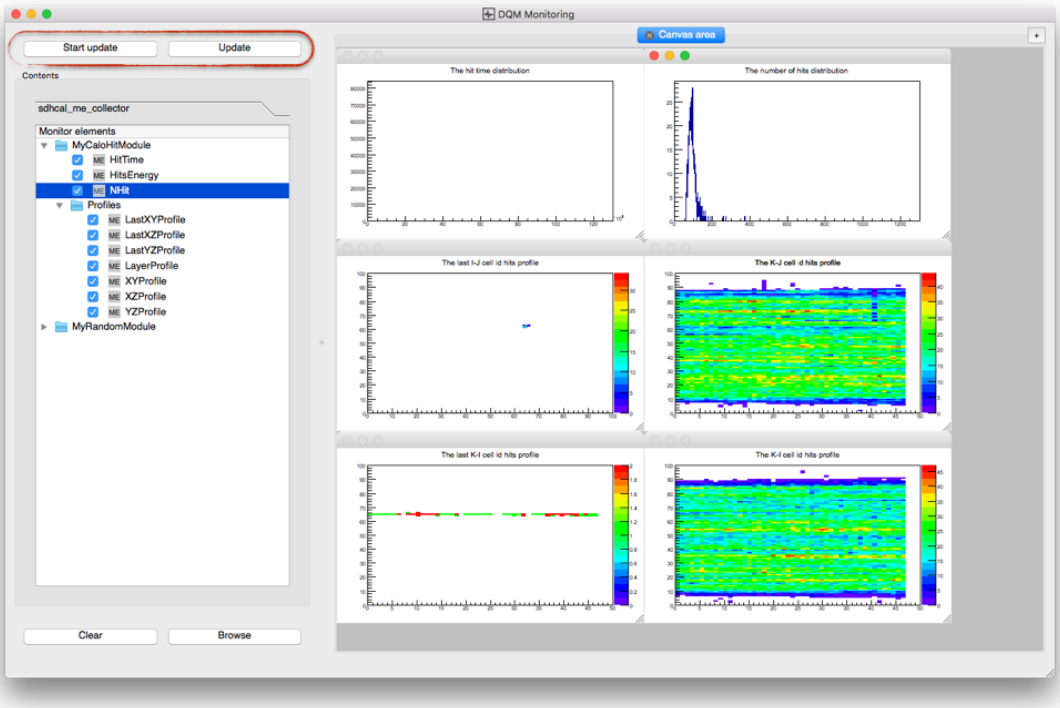

Figure 3.5: An example of the current Qt-based monitoring GUI in use.

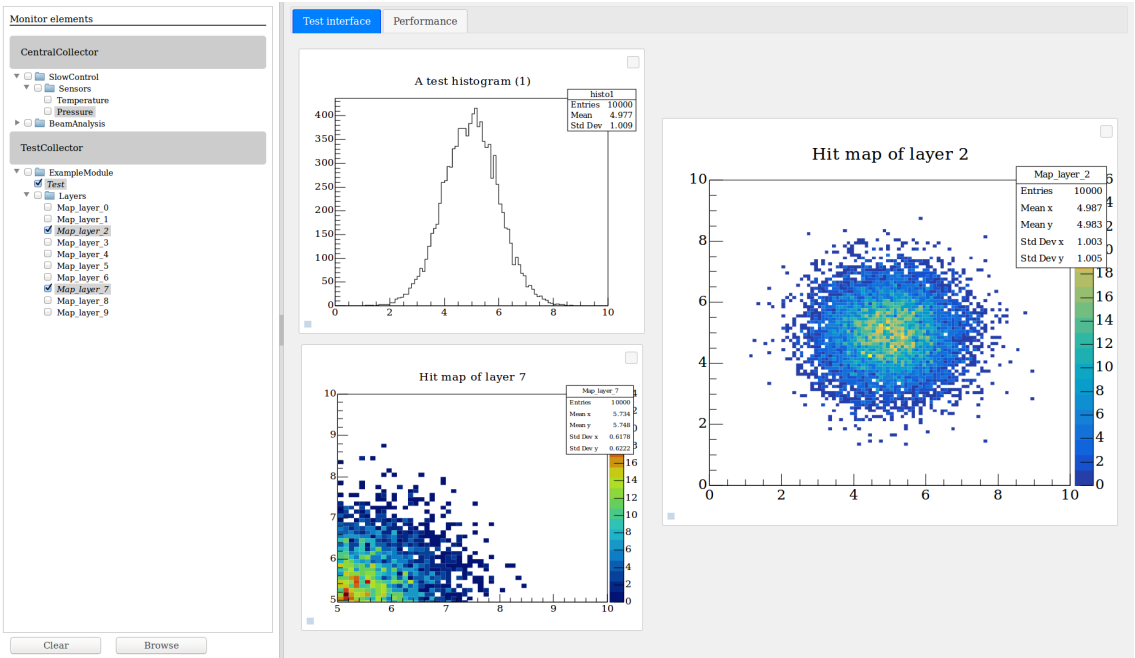

Figure 3.6: A preview of the planned web-based monitoring interface.

mean square 90 (RMS90), or median. It then checks whether either: that this parameter is within a user-specified range; or that it is above or below the user-specified threshold. If a range is being used, the result is the p-value of the property being within the specified range. If a threshold is being used, then the result is 1 if the property passes the threshold, 0 otherwise.

3.2.2 Exact reference comparison test

This is a quality test that takes any TObject, and compares it to a user-specified reference object (which must be of the same type). The result is 1 if the two objects are identical, 0 otherwise.

3.2.3 Fit parameter in range test

This is a quality test that takes either a TH1, TGraph, or TGraph2D object and plots a user-defined function onto it, solving for one of the parameters of the function, then checks it against a user-defined range. The result is the p-value of the parameter being within the specified range.

3.2.4 Kolmogorov-Smirnov test

This is a quality test that takes either a TH1 or a TGraph object, and performs the Kolmogorov-Smirnov test between that object and a specified reference. The result is the p-value of the Kolmogorov-Smirnov test. The Kolmogorov-Smirnov test is intended for unbinned data, not histograms, but ROOT provides a function for performing the Kolmogorov-Smirnov test on histograms, so this is functionlity is also included for the sake of completeness.

3.2.5 χ^2 test

This is a quality test that takes a TH1 object and performs the Pearson χ^2 test between that object and a specified reference. This test is analogous to the Kolmogorov-Smirnov test, but is designed specifically to work for binned histogram data. The result is the p-value output by the χ^2 test.

3.3 Adaptation to other detectors

Due to the modularity and genericity of DQM4hep, the process of deploying it for a new detector is simple – the only parts of DQM4hep that need to be made for any specific use case are the analysis modules, standalone modules, streamer plugin, and file reader plugin. For all of these plugins, there are templates available in the codebase, as well as examples of in-use plugins for other detectors. A few special DQM4hep-specific functions are necessary for these plugins to hook into the framework properly, but apart from these all user-provided plugins are written in normal C++ code that also integrates ROOT, so should be familiar to most users.

A file streamer or file reader must be written by the user, given a specific data structure. This requires knowledge of both the event data model of the data acquisition setup, as well as the structure of the data files. The ideal person to write this code is someone with detailed knowledge of the data acquisition software being used, and the data storage or streaming.

In general, only one of the file streamer or file reader plugins will be needed. If the data will be monitored offline or "nearly-online" by loading files from disk, then a file reader plugin must be written. If the data is to be monitored online, then a streamer plugin must be written. Both of these plugins are similar in structure and differ only on where they get the data from – a file reader loads a file from disk, whereas a streamer loads it from the data acquisition system.

Once the information is accessible from either the file reader or streamer, the framework handles passing this data to the analysis modules. Analysis modules are a type of plugin which take data that has been packaged into events by a file reader or streamer plugin and performs some analysis on it. This usually takes the form of processing the data in histograms, graphs, or some other plot, but can also include an arbitrary amount of other processing, such as checking validation bits, thresholds, error-checking, and so on. Monitoring the data quality can also be done from within analysis modules, but this is not recommended as dedicated quality tests are available.

For each analysis module that is being run, an XML steering file is required to provide the parameters and networking information to all the processes needed. A single steering file can call only a single analysis or standalone module, but multiple steering files can be run in parallel by the framework.

Examples of using DQM4hep with testbeams both within the AIDA-2020 community and outside of it can be found in Chapters 4 and 5, respectively.

3.4 Documentation and user manual

One of the biggest hurdles to promoting a new framework is the lack of understanding of it's installation, deployment, and usage. Many research teams will continue to use their existing software solutions, which may be suboptimal or difficult to use, according to the principle of "better the devil you know than the devil you don't". The first step to overcoming this is to produce clear, readable and complete documentation across the entire range of features the framework has.

To this end, DQM4hep has two sets of documentation: a Doxygen webite and a user manual.

Many particle physics software frameworks provide documentation in the form of Doxygen, a documentation generator tool. Doxygen is extremely useful, as the documentation for functions and objects is written within the code itself, ensuring that documentation is written as the code is written. Doxygen also automatically compiles a documentation guide using HTML, automatically listing the relationships between objects, structures, functions, etc. that can be compiled and viewed locally, or hosted on the internet as a reference.

One limitation with Doxygen documentation is that is not holistic. Doxygen relies upon the structures of the code, not the program as a whole. Doxygen is extremely useful for developers, as well as already-experienced users, but doesn't aid new users in learning to use a piece of software. Provide this more holistic documentation is very important, and would need to take the form of guides and walkthroughs for common procedures, intended for *users* of the framework with little to no interest in the mechanics of it.

To this end, DQM4hep also has a user manual, compiled using Read the Docs, which provides explanations and worked examples of individual components of the framework and running it as a whole. It also includes instructions and details on how to write plugins of all kinds, and how to run the framework both online and offline.

The user manual can be found here [3], and the Doxygen can be found here [4].

Chapter 4

AIDA-2020 testbeams

I love fools' experiments.
I am always making them.

Charles Darwin

One of the most important aspects of testing and developing DQM4hep was to ensure that it was as generic as it was intended to be, and this meant deploying and using the framework on physics testbeams. DQM4hep was developed during testbeams of the SiWECAL, and its early testing phases were predominantly based on this detector, so it was apparent that it could be used in the originally-intended setting. However, in trying to develop it as a generic monitor, and to satisfy the requirements of a generic data monitoring and quality monitoring tool for AIDA-2020, it was essential that it was tested on other detectors of different types to demonstrate its generic natures.

[...]

4.1 Introduction

The CALICE testbeams were done with the CALICE collaboration, working within the *Forschung mit Lepton Collidern* (EN: "Research with Lepton Colliders", or FLC) on the Analogue Hadronic Calorimeter (AHCAL) prototype during its development. The AHCAL is a sampling calorimeter formed of steel absorber plates and plastic scintillator tiles, read out by silicon photomultipliers (SiPMs) as active material. One of the important features of the AHCAL is that the prototypes were assembled using techniques suitible for mass production, such as injection-moulding and automated foil-wrapping of the scintillator tiles, as well as the usage of pick-and-place machines for the assembly of layers and their electronics. It also uses power pulsing to reduce power consumption and heat

production, rapidly cycling power to be active only when the beam is present, according to the known beam structure. [5]

[...]

Regular testbeams were held at the DESY II synchrotron at DESY in Hamburg, Germany and at the Super Proton Synchrotron (SPS) at CERN in Geneva, Switzerland. [...]

[...] DQM4hep was used as the online monitoring and data quality monitoring tool for the AHCAl beginning in May 2016, and in further testbeams between 2016 and 2018. The majority of these testbeams occurred at the DESY II facility, but two took place at the CERN SPS in May 2017 and June 2018.

4.1.1 May 2016 at DESY II

The testbeam where DQM4hep was deployed for another detector for the first time was during a testbeam of the AHCAL at DESY II during May 2016. The testbeam was to be two weeks in duration, with a one-week setup and preparation period. There were a variety of goals for this long testbeam as a whole, including testing some of the data acquisition hardware and software, developing calibration methods, and to test the deployment of DQM4hep. [...]

[...] [This was the three-week one, where we got some real shit done.]

Before and during testbeam, the majority of development for AHCAL-specific analysis modules was undertaken. Prior to this, DQM4hep had only been used on SiWECAL beams, and was untested for other detectors.

However, file reader and streamer plugins for the LCIO data format were already available in the now-deprecated `dqm4ilc` package, which meant that the framework could open and access the data format already.

Data format

The data for the AHCAL was in an LCIO format called LCGenericObject, which is a generic format for use when the existing data formats are not suited. It comprises a header that contains user-defined parameters. These parameters usually include a timestamp, the typename of the object, and a string describing the content of the data and how to parse it. For instance:

In this case, the `TDC14bit[NC]` and `ADC14bit[NC]` are arrays, each holding a number of elements equal to the `NChannels` variable, in this case 36. Each element of these arrays corresponds to a single physical scintillator tile within the detector, and identifies which

chip it belongs to using `ChipID`.

[...]

Analysis modules

To begin with, two analysis modules were developed:

- `AHCALChannelSpectra` created a histogram for each channel present in the detector, and filled it with the ADC value for each readout cycle in the whole run, producing a per-channel spectrum of ADCs.

- `AHCALHitmap` created a two dimensional histogram, with each bin representing a channel on the x and y axes, and filled each channel with the ADCs of that channel for the whole event, producing a hitmap.

[...]

Creating the hitmap was nontrivial, as the information coming from the data acquisition and stored in SLCIO format did not have geometric information for each channel, instead only specifying the "electronics number" – a combination of the board number and channel number. To determine the location of any given channel, a mapping was needed. This mapping was not simple – each board contained sixteen channels, and each layer was formed of four boards tiled together. Further, the numbers of the boards and which order the layers were stacked could also change.

For every testbeam, there was a mapping file that described the position of each board and channel in (i, j, k) co-ordinates. To implement the hitmap analysis module, the module used DQM4hep's libraries for XML parsing to build two functions inside the analysis module class – `electronicsToIJK` and `IJKToElectronics` – that converted either from electronics number to geometric co-ordinates, or vice versa.

Using this method ensured that, given the geometry file was provided, whatever changes in geometry or set-up of the experiment were always reflected in the hitmaps, and no alteration of hard-coded information or recompilation of the modules was required.

[...]

Results

[...]

4.1.2 December 2016 at DESY II

[...]

4.1.3 May 2017 at CERN SPS

During May 2017, testbeam time at CERN's Super Proton Synchrotron (SPS) facility was used for further tests for the AHCAL.

[Figure: we have plenty of pictures of the testbeam area and the installation.]

[...]

During this testbeam, the analysis modules were mature, and represented the majority of the needed online monitoring for the testbeam, especially because the data format of the detector had been fixed for some time. Because of this, after the initial set-up and verification stages, very little management or editing of the monitoring software was necessary. It was instead used as intended – a tool for shifters to use to troubleshoot problems with the beam or detectors. It was successfully used to identify dead channels on several of the boards [confirm this].

[...]

[Plots from the AHCAL CERN testbeam]

Chapter 5

IDEA testbeams

New quote split
over two lines

Author McWriter

While DQM4hep was intended as a generic tool, it was largely developed within the AIDA-2020 collaboration, which also promoted a variety of standardisations for data acquisition devices, data formats, etc. In order to test whether DQM4hep was truly generic, and able to adapt to any type of detector, it needed to be tested outside of the AIDA-2020 groups and facilities. An ideal opportunity was presented for this, with a collaboration of Italian universities working on a number of particle detectors, hoping to do a single combined testbeam at the CERN SPS.

[...]

5.1 Introduction

The combined testbeam took place between 5th-12th September 2018 at the CERN SPS beamline facility. [...]

5.1.1 Detectors present at the combined testbeam

The combined testbeam comprised four separate dectors: a calorimeter, a muon detector and preshower, a drift chamber, and a silicon photomultiplier. One of the biggest challenges involved in the testbeam was operating these four different detectors [...]

RD52 calorimeter

The calorimeter was formed of two layers of 36 tiles each, totaling 72 tiles, stacked behind each other. One layer used Cherenkov detectors, the other used scintillator tiles. In addition, there was a group of leakage detectors that detected whether individual events were contained within the calorimeter or not. [DWC - Delayed Wire Chamber?]

Muon chamber and preshower

[...]

Silicon photomultiplier GEM

[...]

Drift chamber

[...]

5.2 Progression of the testbeam

[...]

Existing monitoring within the DREAM collaboration could produce accurate histograms from raw data using ROOT, creating plots of the energy spectra of each detector channel per event, along with [other things]. This facility was reproduced in DQM4hep quickly using for-loops in both the C++ code and XML steering files, allowing this to be done with comparatively little code.

5.2.1 File readers

[...]

The RD52 calorimeter, drift chamber, and muon and preshower were all read using ROOT ntuple files, generated by the DAQ one step after reading and saving the data in a raw binary format. This made the file reader code simpler, faster and more readable, as it only needed to walk through ROOT trees and extract data from leaves event-by-event, rather than reading from raw binary or hex data.

For the silicon photomultiplier GEM data, the "raw" data format was a text file, containing an XML header followed by a large amount of data in comma-separated values (CSV). This file could be loaded directly into DQM4hep, the XML header separated and

parsed with DQM4hep's internal XML parsing libraries, and the remaining data parsed. The comma-separated values could be easily parsed using the `dqm4hep::core::tokenize` function, which takes a string, a delimiter, and a vector, and parses the string into values separated by the delimiter, loading them into the vector. This made extracting the GEM data extremely simple, even in this format.

5.2.2 Analysis modules

[...]

5.3 Results

[...]

5.3.1 ADC to energy calibration

[...]

This required several runs whose characteristics were similar or identical, except for the energy of the beam. The runs chosen where run numbers [...], summarised in Table [ref].

A plot was made of the average ADC per event summed over all channels, against the beam energy for that run. See Fig. [ref]. Then using a linear least squares fit, the equation that describes the transformation from ADC to energy was found.

[...]

5.3.2 Tower ADC calibration

[...]

Due to a large number of tasks for setting up the testbeam, performing a calibration of the individual towers' high voltages was not possible, so the calorimeter ADCs were not calibrated to each other [?]. In order to fix this,

Two sets of calibration runs were performed. The first set covered Towers 1-29 using an 80 GeV secondary beam (π and e^-), with the beam pointed at each tower in sequence for 29 runs. The second set used a 20 GeV electon beam, covering Towers 30-36 plus Tower 15. Tower 15 recieved runs in both sets in order to calibrate the two sets to each other.

[...] The run numbers corresponding to each tower is given in Table 5.1.

Tower	Run No.	Tower	Run No.
1	12545	16	12526
2	12556	17	12567
3	12558	18	12633
4	12560	19	12591
5	12601	20	12612
6	12638	21	12530
7	12598	22	12528
8	12514	23	12569
9	12518	24	12639
10	12521	25	12610
11	12600	26	12609
12	12636	27	12607
13	12539	28	12604
14	12628	29	12602
15	12512		

Table 5.1: Table of the run numbers and corresponding tower numbers for the calibration runs.

The process for calibrating the towers was to make a histogram of the ADCs registered in a tower, only in the run where the beam was centered on them, in order to find the mean and standard deviation. These histograms were then combined to show the overall responses of the entire set of calibration runs, which visualises the differences between the Towers. These can be seen in Fig. 5.1.

Since Tower 15 was present in both runs, this was chosen as the reference, and all the other towers were given a coefficient that leveled their mean ADC to the same value as Tower 15, in both sets of calibration runs. The ADC values for each tower are then multiplied by their calibration coefficient. [...] [Difference between scintillator and Cherenkov] [...] The calibration coefficients are shown on Table 5.2.

5.3.3 Particle selection efficiencies

[...]

Tower	Coefficient	Tower	Coefficient
1	x	16	x
2	x	17	x
3	x	18	x
4	x	19	x
5	x	20	x
6	x	21	x
7	x	22	x
8	x	23	x
9	x	24	x
10	x	25	x
11	x	26	x
12	x	27	x
13	x	28	x
14	x	29	x
15	x		

Table 5.2: Table of tower numbers and their calibration coefficients.

Using the ancillary detectors

The RD52 calorimeter was designed with a set of ancillary detectors to help discriminate between particle of different types. These provide a first selection of different particles, allowing us to use kinematic properties from within the calorimeter itself to perform a second particle selection, and then compare the two. [...]

The two ancillary detectors used for particle selection are the [] muon trigger. [...]

[...]

Using the calorimeter

[...]

One of the first steps is using the RD52 calorimeter data to find the energy ratio R for each event:

$$R = \frac{E_1}{\sum_{i=1}^{n} E_i}$$

where E_i is the energy of the i^{th} most energetic channel in the event and n is a nonzero integer. The choice of n [...]. Once the ratio R is calculated, a plot can be made of E_{total}

Figure 5.1: The plot of ADCs for each tower before calibration.

vs. R for an entire run that shows separation of electrons from muons and pions – see Fig. 5.3.

An appropriate cut can be used to select electron events. [Information about the cut]. Adding in the information from the RD52's muon trigger, muons and pions can then also be separated. Using both the cut and the muon trigger, we can thus produce spectra for each individual type of particle in the run.

[...]

The main goal of the data analysis is to characterise the response of the detector, and to measure the selection efficiencies of the detectors to various particle types. Once this is done, this will also give us a detailed account of the beam composition during each of the runs, which can be used for further work.

In order to do this, several ways to select for different particle types are necessary. The first way was using the preshower detector and muon trigger, which are both designed to discriminate between electrons and muons (respectively), with a high selection efficiency. These are used to create "reference" samples, [...]

The second way is to perform a kinematic selection using variables from the calorimeter. This is done using the E vs. R plot (normalised for beam energy) to select a region corresponding to a certain particle type. For example, the plot below shows this plot for Run [xxx] (X GeV hadrons) with the regions corresponding to hadrons and electrons highlighted. These regions overlap, meaning that attempting to select for hadrons using an ellipse around that region will also result in a non-insignificant number of electrons also

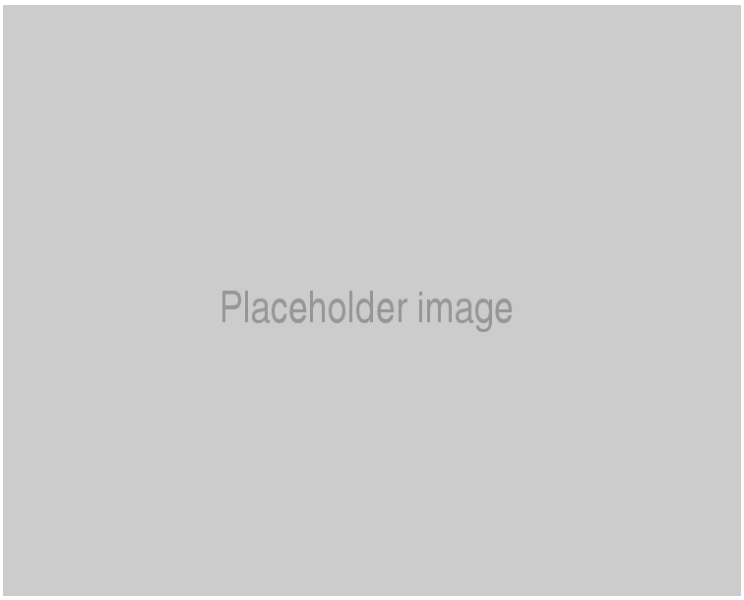

Figure 5.2: Comparison of the calibration plots for the ADCs before and after calibration.

being selected.

In order to perform a pure selection using the calorimeter, an extremely tight cut was used, focusing on the red spot at the centre of the hadron region. This ensures that the majority of events that pass the cut are hadrons, giving a high-purity selection. The purity of this selection can be assessed by using the appropriate preshower or muon trigger, excluding all non-hadron particles, and comparing the two numbers. If N_K is the number of particles passing the selection with only the kinematic cut, and N_{K+T} is the number of particles passing *both* the kinematic cut and the triggers, then the selection efficiency for hadrons ϵ_{hadron} is given by:

$$\epsilon_{hadron} = \frac{N_{K+T}}{N_K}$$

This was then done with the same process for electrons and muons, to obtain the individual selection efficiencies.

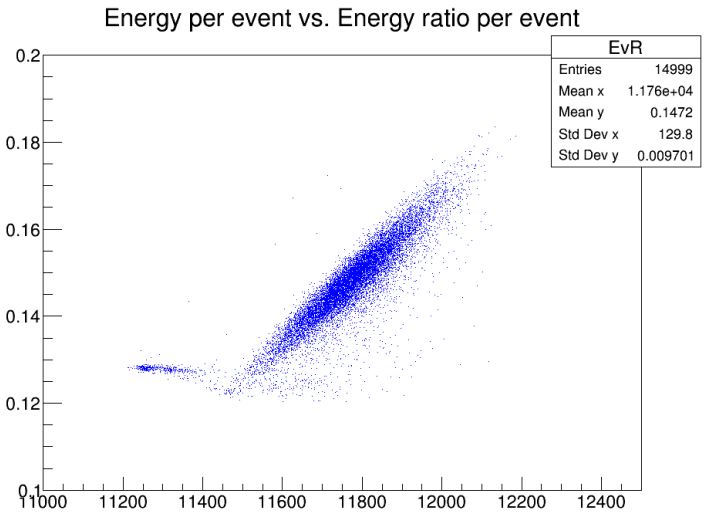

Figure 5.3: Plot of E against R for the secondary hadron beam with $n = 10$ (Run 12709).

Chapter 6

Physics studies for the Compact Linear Collider

Somewhere, something incredible is
waiting to be known.

Carl Sagan

One of the primary goals of future lepton colliders like the ILC and CLIC is to become "Higgs factories" – machines that can produce large numbers of Higgs bosons in a variety of final states, allowing the Higgs sector of the Standard Model to be probed with unprecedented accuracy and coverage.

One of the uniquely accessible measurements for these colliders is a precision measurement of the top-Higgs Yukawa coupling. This serves as a further test for the Standard Model and [...]

Another important avenue to pursue is CP-violation in the Higgs sector. Since Higgs physics is still an emerging field, it is not yet known whether CP-violation is present in the Higgs sector to the degree that the Standard Model predicts. It is also a fertile area for investigation of BSM physics, as many BSM models predict additional Higgs bosons, or Higgs bosons with characteristics that differ from the SM Higgs.

The $e^+e^- \to t\bar{t}h$ event (see Fig. 6.1) is one process that is both accessible to CLIC's design energy and extremely useful for interrogating the Higgs sector for CP-violation and BSM physics. The production of Higgs bosons allows for several observables that would be sensitive to any Higgs bosons with an odd CP quantum number (or "CP-odd" Higgs bosons). Determining the detectors' sensitivity to the ratio of CP-odd and CP-even Higgs bosons (also called the CP mixing angle) will allow further understanding of the limits of the Standard Model, as well as the limits on the various BSM physics models, and regions

of interest for possible new physics.

[...]

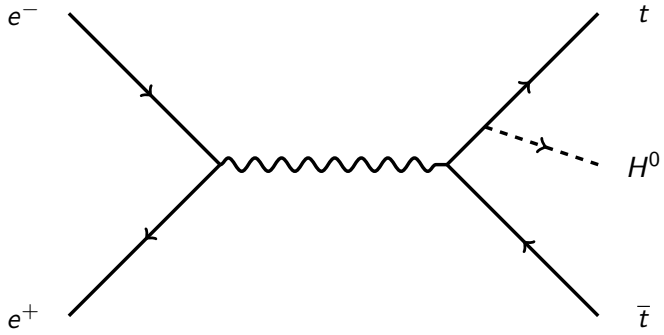

Figure 6.1: A Feynman diagram of the tth event.

There are three final states of the tth event, which depend on the decays of the $W^{\pm}$ boson. The $W^{\pm}$ can decay into either a quark-antiquark pair, or a lepton-neutrino pair, so there are three possible final states: the *fully hadronic* case, where both $W^{\pm}$ particles decay into quark pairs; the *leptonic* case, where both decay into lepton-neutrino pairs; and the *semi-leptonic* case, where one decays into a quark pair and the other into a lepton-neutrino pair. In general, the leptonic final state is not utilised for this analysis, so is not discussed further. Extended Feynman diagrams of the fully hadronic and semi-leptonic final states are shown in Figs. 6.2 and 6.3

[...]

[...] The invariant mass of the Higgs boson can be determined by summing the invariant masses of pairs of bottom quarks and computing the χ^2 for each possible combination; the combination with the lowest χ^2 shows the pair that has decayed from the Higgs boson.

[...]

6.1 Physics generation and samples

[...]

The Monte Carlo samples were generated predominantly using Whizard 1.95, though for the Higgs events, Physsim was used due to technical contraints of Whizard [ref?]. All samples were simulated at $\sqrt{s} = 1.4\,TeV$ and unpolarised beams, assuming an integrated luminosity of 1.5 ab^{-1} and a light Standard Model Higgs boson with mass 125 GeV/c^2. See Table 6.1 for a summary of all of the used samples. The first two rows are the ttH signal channels, all other rows are background. The number of jets refers to the number of jets in the final state that have come from the decay of the top-quark pair. The number of

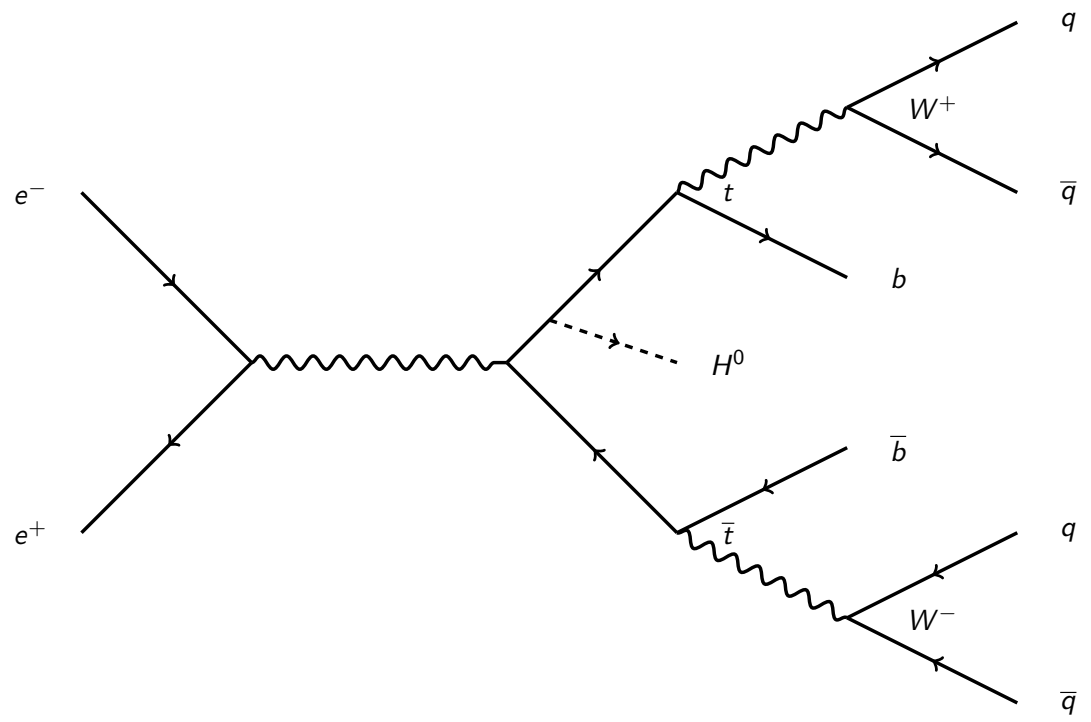

Figure 6.2: An extended Feynman diagram of the tth event, showing the fully-hadronic decay channel with the final state $q\overline{q}q\overline{q}b\overline{b}$, where q and $\overline{q}$ indicate a quark-antiquark pair.

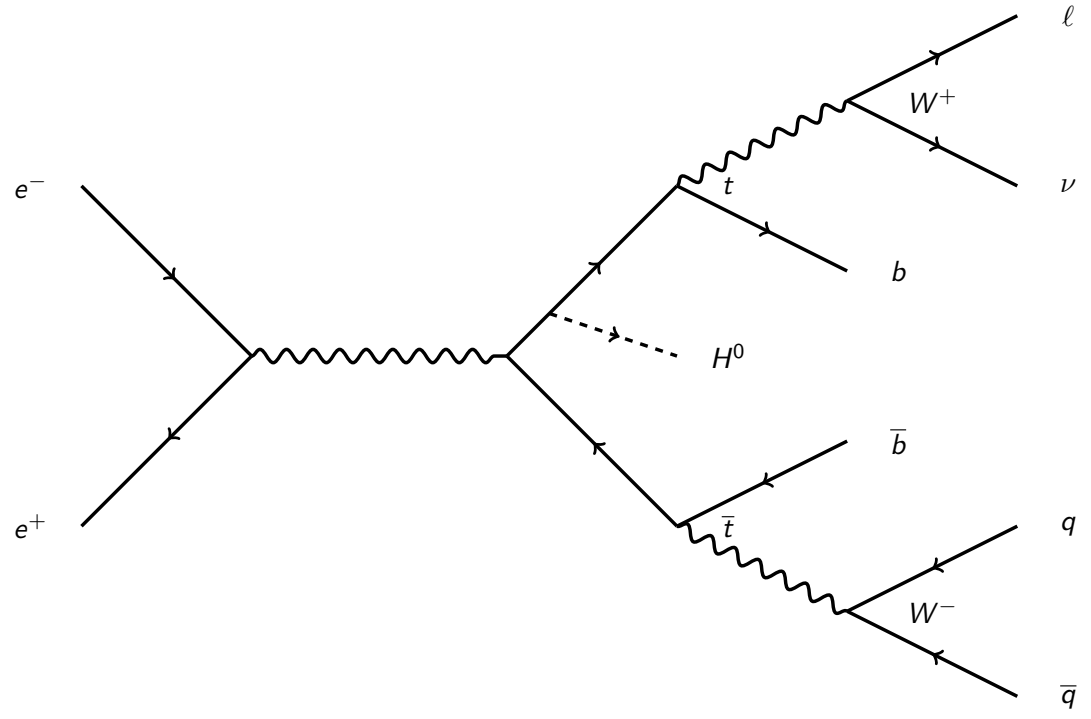

Figure 6.3: An extended Feynman diagram of the tth event, showing the semi-hadronic decay channel with the final state $\ell\nu q\overline{q}b\overline{b}$, where ℓ and ν indicate a lepton-neutrino pair of the same flavour but opposite [sign?].

events in 1.5 ab^{-1} has been calculated from the integrated luminosity and sample weight.

ProdID	Process	Cross-section (fb)	Sample weight	Events in 1.5 ab^{-1}
2435	$t\bar{t}H$, 6 jets, $H \to b\bar{b}$	0.431	0.03	647
2441	$t\bar{t}H$, 4 jets, $H \to b\bar{b}$	0.415	0.03	623
2429	$t\bar{t}H$, 2 jets, $H \to b\bar{b}$	0.100	0.006	150
2438	$t\bar{t}H$, 6 jets, $H \not\to b\bar{b}$	0.315	0.02	473
2444	$t\bar{t}H$, 4 jets, $H \not\to b\bar{b}$	0.303	0.02	455
2432	$t\bar{t}H$, 2 jets, $H \not\to b\bar{b}$	0.073	0.004	110
2450	$t\bar{t}Z$, 6 jets	1.895	0.1	2843
2453	$t\bar{t}Z$, 4 jets	1.825	0.1	2738
2447	$t\bar{t}Z$, 2 jets	0.439	0.03	659
2423	$t\bar{t}b\bar{b}$, 6 jets	0.549	0.03	824
2426	$t\bar{t}b\bar{b}$, 4 jets	0.529	0.03	794
2420	$t\bar{t}b\bar{b}$, 2 jets	0.127	0.008	191
2417	$t\bar{t}$	125.8	1.5	203700

Table 6.1: Table of all signal and background samples used for this analysis.

6.2 Detector models

[...]

[...] henceforth referred to as CLIC_SiD.

6.3 Sensitivity to cross-sections and Yukawa coupling

[...]

6.3.1 Analysis method

[...]

Sample processing

The first step is the initial jet clustering. This is done using the k_t algorithm with parameters [?], using an exclusive clustering mode to form 8 jets – 6 jets from the produced quarks and 2 beam jets. The k_t algorithm is used over choices like anti-k_t and Valencia, because

the important features are the *relative* shapes of the jets, rather than absolute properties, so there is no need to use more computationally-intensive [is this true?] algorithms.

Once initial jet clustering is finished, a Marlin processor that finds isolated leptons is used. It searches for either 0, 1, or 2 isolated leptons, and this information can be used to make decisions about whether to process the event already:

Leptons	Channel	Action
0	Fully hadronic	Use for fully hadronic analysis
1	Semi-leptonic	Use for semi-leptonic analysis
2	Fully leptonic	Discard

Following this, the two beam jets are removed from the processing, and a further step of jet re-clustering is performed, using the Durham algorithm, to [...]

[...] [Flavour tagging]

[...] [Tau finding]

The final step is to use use PandoraPFAs to generate Particle Flow Objects (PFOs) of the undetectable particles, especially the top quarks, $W^{\pm}$, and Higgs. [...]

Analysis processing [?]

Once the sample has been processed, it must be analysed. The first step of this is a program used to extract various kinematic variables of both particles in the events (m_0, p_t) and the event itself (Ψ, T). This is the Treemaker program, and [...] [Chi-squared extraction of invariant masses] [Feeding into TMVA to generate BDTs]

[...]

A flow diagram of the analysis process, and the rejection points, is shown in Fig. 6.4.

[...]

6.3.2 Results

[...]

The combined uncertainty for the cross-section of both decay channels is:

Cross-section:

$$\Delta\sigma = 7.30\%$$

Then using this and a linear approximation from QCD [ref], the value of the uncertainty on the top-Higgs Yukawa coupling can be computed:

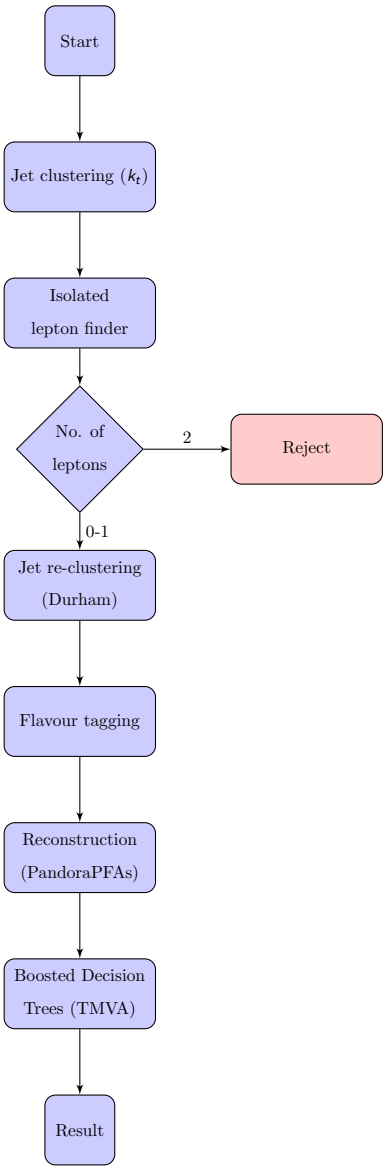

Figure 6.4: A flow diagram of the algorithm for analysis of ttH events, and rejection.

$$\frac{\Delta g_{tth}}{g_{tth}} = 0.503 \frac{\Delta\sigma(t\bar{t}H)}{\sigma(t\bar{t}H)} = 3.86\%$$

These results were contributed to a paper that summarised the top physics potential for CLIC at $\sqrt{s} = 1.4$ TeV, published in [journal][ref] and will be submitted to CERN's European Strategy Update in [month] 2019 [ref].

6.4 Determination of sensitivity to CP-violation

[...]

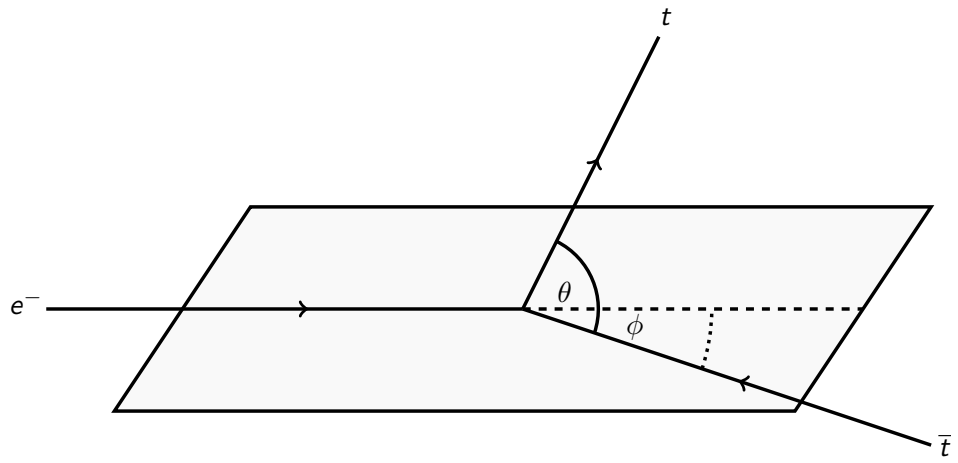

Figure 6.5: Geometric diagram of the up-down asymmetry in tth events. The paths of the electron and antitop quark, and the angle ϕ between them define a plane. "Up-going" top quarks go above the plane, "down-going" top quarks go below.

6.4.1 CP-sensitive observables

[...]

Up-down asymmetry

The up-down asymmetry is a conceptually simple observable that has already been identified for investigating CP-violation in the tth process. It is found by defining a plane from the vectors of the incoming electron and produced antitop, then finding the ratio of top quarks that are emitted above and below this plane (see Fig. 6.5). If there is no CP-violation in this process, the ratio will be even.

Using the up-down asymmetry as an observable requires that the W^+ and W^- can be distinguished from each other, and has thus far only been used for the semi-leptonic decay channel. In this case, the lepton produced by the decay of one of the W bosons identifies its charge, and thus the charge of the top quark that it has decayed from. While this method was not previously possible in the fully hadronic case, a method for applying it by using jet charge determination is discussed in Section 6.4.2.

[other ones]

[...]

6.4.2 Jet charge determination

Previous analyses that have utilised the up-down asymmetry as an observable have focused exclusively on the semi-leptonic decay channel of tth events, as the presence of a lepton emitted by the top or antitop quark offers a simple and statistically robust method to distinguish between the two top quarks. In the hadronic decay channel each top emits a jet, and even in the ideal case where each particle resulting from the jet can be accurately reconstructed, the net charge will *still* be an integer, since no particles with a non-integer charge can result from these decays.

However, techniques developed in recent years, intended primarily for observations in ATLAS at the LHC, have refined methods for this, and using the work of [reference], it is now possible to obtain the total net charge of the jet – that is, the charge of the initial quark that creates the jet.

This technique is strongly-dependent upon the accuracy and efficiency of both particle reconstruction and jet clustering, but these techniques are constantly improving, and Pandora Particle Flow Algorithms (PFAs) and new jet clustering methods are becoming increasingly sophisticated. Combined with the cleaner final states in a lepton collider, these techniques allow the charge of a jet to be determined with useful confidence.

The charge of a jet can be determined by summing the charges of all particles in the jet, weighted by p_T and normalised by the p_T of the entire jet:

$$Q_\kappa^j = \frac{1}{(p_T^{jet})^\kappa} \sum_{j \in jet} Q_j (p_T^j)^\kappa$$

Where κ is some parameter between 0 and 1, typically set to 1. With this technique, it is possible to determine between quarks with charges of $+1/3e$, $-1/3e$, $+2/3e$, and $-2/3e$ with [some level of confidence].

Jet clustering algorithms

Since this method relies upon jets it is strongly dependent upon jet reconstruction, and thus on the choice of jet clustering algorithm and parameters. Previous analyses of tth events have used a two-step reclustering approach, using the k_t algorithm for the initial clustering and the Durham algorithm for reclustering. These algorithms were chosen as the relative difference between the jet shapes is more important than their absolute shapes, so other algorithms do not provide any benefits.

The Valencia algorithm, however, gives improved performance in the cleaner final states of a lepton collider, for which it was especially designed, and many analyses are

now transitioning to using the Valencia algorithm.

6.4.3 Results

[...]

Chapter 7

Discussion and Conclusions

What we know is really very, very little
compared to what we still have to know.

Fabiola Gianotti

[...]

Bibliography

[1] Assessment of the revised plan of international linear collider project (executive summary). http://www.scj.go.jp/ja/info/kohyo/pdf/kohyo-24-k273-en.pdf, 2018. Accessed: 2019-02-07. 3

[2] R Ete, T Coates, and A Pingault. Dqm4hep: a generic data quality monitoring framework for hep. Technical report, 2018. 7

[3] Dqm4hep user manual. https://dqm4hep.readthedocs.io/en/latest/. Accessed: 2019-02-07. 16

[4] Dqm4hep doxygen pages. https://dqm4hep.github.io/dqm4hep-doxygen/doxygen/dqm4hep/master/index.html. Accessed: 2019-02-07. 16

[5] Felix Sefkow and Frank Simon. arxiv: A highly granular sipm-on-tile calorimeter prototype. Technical report, 2018. 18

Appendix A

Code

```
10 PRINT "LOOK AROUND YOU"
20 GOTO 10
```