UNIVERSITY
OF SUSSEX

Data acquisition software development and physics studies for future e^+e^- colliders

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 future e^+e^- colliders

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 Colliders

Progress is not a straight line.

An Wang

In the post-LHC era, particle physics is at somewhat of an impasse. The Standard Model (SM) has held up to most experiments and observation, and its predictive power is now exhausted. There are tantalising hints at a theory beyond the Standard Model – in the form of CP-violation and dark matter – but as of yet all new ways to probe the SM for its weaknesses, to see the greater theory behind it, have yielded very little.

There are many planned investigations to attempt to identify physics Beyond the Standard Model that use the Large Hadron Collider (LHC), or plan to leverage the upgrades for the High Luminosity Large Hadron Collider (HL-LHC). However, now that the Higgs boson has been identified successfully, one of the most fruitful avenues for further research is the construction and operation of a lepton collider at the energy frontier, with sufficient centre-of-mass energy to produce Higgs bosons in large numbers.

This is what motivates the several proposals for future lepton colliders around the world today, that would operate to complement and expand the reach of particle physicists beyond what the LHC is currently capable of. These proposed lepton colliders are broadly split into two groups: linear colliders and circular colliders.

Linear colliders use two accelerator arms pointed towards a single interaction point, and in general are capable of high centre-of-mass energies thanks to developments in accelerator technologies. The main candidates for this style of collider are the International Linear Collider (ILC) and the Compact Linear Collider (CLIC).

Circular colliders use a similar layout to the LHC, with a circular accelerator capable of creating collisions at multiple different interaction points around the circumference of the ring, which permits multiple detectors. The downsides of a circular collider are that

lighter particles like electrons are more prone to energy losses via synchrotron radiation, limiting the centre-of-mass energies that a circular lepton collider can operate at. However, in exchange they tend to have much higher luminosities, allowing greater numbers of collisions and higher yields of certain processes or channels. The main candidates for circular colliders are the Future Circular Collider (FCC) and the Circular Electron Positron Collider (CEPC).

These proposed colliders share many of the same motivations, design considerations, features, and challenges, as does the ongoing international effort in research and development to make these colliders a reality.

2.1 The physics case for a lepton collider

With the observation of a Higgs boson with a mass of 125 GeV, based on data from the Large Hadron Collider, the Standard Model of particle physics is now functionally complete. All of it's major predictions have been observed. This is a testament to it's quality as a theory, where it's predictive power and accuracy is one of the best in all of the sciences.

But despite this, a multitude of observations have shown that the Standard Model cannot be a complete theory of nature. Many phenomena have been observed that the Standard Model cannot predict, or do not interact with the Standard Model in any way. In the post-LHC era, particle physics is now forced to seek answers to three questions that the Standard Model cannot provide answers to:

- What is dark matter? Astrophysics observations support the existence of a neutral, weakly-interacting substance that composes around 85% of all mass in the universe. Yet this substance cannot be explained by any known form of matter, and is completely unprecedented by the Standard Model.

- Why is there so little antimatter? The symmetries inherent in the Standard Model predict that the Big Bang would have created an equal quantity of matter and antimatter. Yet the universe today is dominated by matter.

- Why does the Higgs field fill space and give mass to elementary particles? The existence of the Higgs field and the coupling of the Higgs boson to other particles can be understood from the Standard Model but their origin or cause is still unexplained.

In order to answer these questions, new theories of physics Beyond the Standard Model

(BSM) have been made, and need to be experimentally tested. To do this, particle collider experiments at the energy frontier are needed. The Large Hadron Collider (LHC) has already been used extensively for searches for new particles, rare and exotic decays, supersymmetry and dark matter.

However, the running of a lepton collider at the energy frontier would be complementary to the LHC's continuing physics programme – there are many events or channels that are inaccessible or difficult to examine in one environment that are much simpler or higher precision in the other. In this way, a lepton collider would help to improve and refine measurements already taken at the LHC, while also allowing physicists to examine new channels and decays that were not accessible to it.

This follows a historical pattern in particle physics – as the energy frontier advances, hadron colliders are used to discover new physics and new phenomena, followed by lepton colliders to examine these phenomena in higher precision.

2.1.1 Higgs physics

Of specific interest to searches at lepton colliders would be the Higgs boson itself. Many BSM models predict differences from the Standard Model in the Higgs sector – such as several Higgs bosons with different masses, composite Higgs, charged Higgs etc. The comparatively 'quiet' environment of a lepton collider allows higher precision measurements of the properties of the Higgs boson, placing better constraints on the presence of new physics. In addition, lepton colliders can easily operate at specific thresholds and "hot spots" for Higgs production, permitting a much greater number of events yielding Higgs bosons, and thus a greater sample to examine.

Additionally, the most common decay of the Higgs boson, at a branching ratio of 57.7%, is the $H \to b\bar{b}$ process. Despite the high branching ratio, the huge QCD backgrounds in a hadron collider have made this decay incredibly difficult to observe at the LHC – in fact, the $H \to b\bar{b}$ decay has only fairly recently been experimentally verified by the ATLAS experiment. However, in a lepton collider these QCD backgrounds are significantly smaller, meaning this decay channel is much easier to analyse, opening up a huge number of Higgs events for analysis and examination.

Another highly interesting process uniquely accessible to lepton colliders is the higgstrahulung reaction: $e^+e^- \to Zh$. The decay of the Z boson into lepton pairs e^+e^- or $\mu^+\mu^-$ allows high precision kinematic measurements of the process without directly measuring the Higgs boson itself. This allows unprecedented measurement of the Higgs mass,

while also making measurements of missing energy possible. If the Higgs boson has invisible decays – such as dark matter particles or other undiscovered particles that don't couple to the SM – the higgstrahlung process allows their existence to be identified.

2.1.2 Top physics

[...]

Energy	Reaction	Physics goal
91 GeV	$e^+e^- \to Z$	ultra-precision electroweak
160 GeV	$e^+e^- \to WW$	ultra-precision W mass
250 GeV	$e^+e^- \to Zh$	precision Higgs coupling
350-500 GeV	$e^+e^- \to t\bar{t}$	top quark mass and couplings
	$e^+e^- \to WW$	precision W coupling
	$e^+e^- \to \nu\bar{\nu}h$	precision Higgs coupling
500 GeV	$e^+e^- \to f\bar{f}$	precision search for Z'
	$e^+e^- \to t\bar{t}h$	Higgs coupling to top
	$e^+e^- \to Zhh$	Higgs self-coupling
	$e^+e^- \to \widetilde{\chi}\widetilde{\chi}$	search for supersymmetry
	$e^+e^- \to AH, H^+, H^-$	search for extended Higgs states
700-1000 GeV	$e^+e^- \to \nu\bar{\nu}hh$	Higgs self-coupling
	$e^+e^- \to \nu\bar{\nu}VV$	composite Higgs sector
	$e^+e^- \to \nu\bar{\nu}t\bar{t}$	composite Higgs and top
	$e^+e^- \to \tilde{t}\tilde{t}^*$	search for supersymmetry

Table 2.1: Physics processes of interest at lepton colliders up to 1 TeV.

[...]

2.2 The International Linear Collider

The International Linear Collider (ILC) is a proposed high-luminosity linear electron-positron collider based upon 1.3GHz superconducting radio frequency (SCRF) accelerating technology. The centre of mass energy ($\sqrt{s}$) would be 250 GeV, upgradable to 500 GeV and then to 1 TeV at a later date. The total footprint of the complex would be 31km in length, with the arms using magnets with an accelerating gradient of 31.5 MVmˆ1 in metre-long superconducting nine-cell niobium cavities operating at 2K.

The ILC is also planned to operate on an innovative "push-pull" system of detectors – two detectors are planned, called the International Linear Detector (ILD) and Silicon Collider (SiD), which would be mounted on a movable base. The detector hall would be large enough to accommodate both detectors such that they could be interchanged.

There were a number of proposed sites for the ILC, including CERN in Geneva, DESY in Hamburg, and JINR near Moscow. The most recent country to seek to host the collider was Japan, who proposed a greenfield site located in the Kitakami Highlands region of Iwate prefecture.

However, a report from the Science Council of Japan (a representative organisation of the Japanese science community) released in early 2019 expressed that they had 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 whether the expected scientific outcomes would justify the unprecedented human resource requirements and infrastructure necessary to make the ILC a reality [1].

On 7th March 2019, the Japanese government expressed that it would not make a proposal to host the collider in Japan. The Japanese government did however express interest in the ILC, and declared that it would be continuing discussion and interest in the project as a whole.

[...]

2.2.1 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)

[...]

The finished ILD will weigh 14,000 metric tonnes, and using a magnetic field of 3.5 Tesla will offer a spatial resolution of 3 microns in the vertex tracker, and 60 microns in

the central TPC tracker.

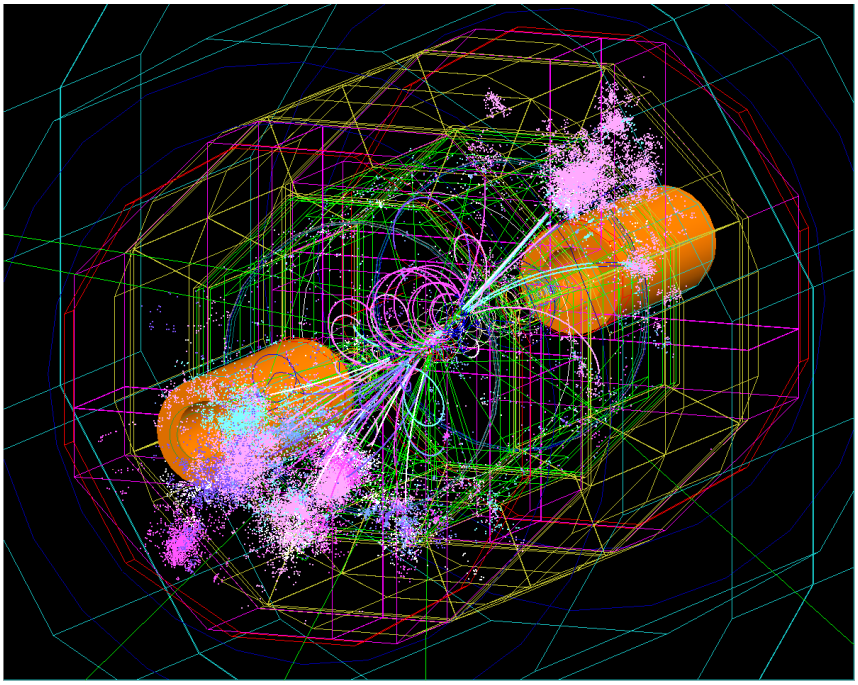

Figure 2.1: Visualisation of a simulated tth event in the ILD. Charged particles can be easily identified by their curved, coiled or spiral paths, and the jets are clearly visible as the light pink and purple areas near the beampipes on either side.

The Silicon Detector (SiD)

[...]

2.3 The Compact Linear Collider

[...]

The CLIC project foresees a programme spanning 22 years, over which multiple upgrades to the centre-of-mass energy would take place. Initial construction would be at 380 GeV, focusing on precision measurements of top-quark and Higgs physics. Further upgrades would increase the centre-of-mass energy to 1.5 TeV, then finally 3 GeV. Physics goals in these later stages would involve searches for new physics processes, as well as precision measurements of rare Higgs processes, and of new states discovered at the LHC or earlier stages of CLIC.

[...] 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. This

placement is determined by the geological features of the region around Geneva and the feet of the Juras, [...]

[...]

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 this to 1.5 TeV and finally 3 TeV.

2.4 The Future Circular Collider

The Future Circular Collider (FCC) is a series of concepts for a future collider that would be located in the Geneva area near the existing LHC ring. The FCC project as a whole has three different accelerator concepts – the FCC-hh for proton/proton and ion/ion collisions, the FCC-ee for electron/positron collisions, and the FCC-he for electron/proton collisions.

The initial proposal is to construct a circular electron/positron collider – the FCC-ee – with a circumference of 100km and delivering a maximum centre-of-mass energy of 365 GeV. The motivation for this is that at this energy range – the electroweak scale – the FCC would be able to access the Z pole, the W- and top-pair production thresholds, as well as producing a large number of Higgs bosons.

A further part of the proposal for the FCC is that following the conclusion of the physics programme of the FCC-ee, the tunnels constructed to house the accelerator would be re-used for the FCC-hh, a hadron collider. This follows a similar use case as the LHC, which re-purposed the tunnels used for the Large Electron Positron Collider (LEP). It is claimed that the FCC-hh built in these tunnels would be able to reach centre-of-mass energies of at least 100 TeV.

According to the given timeline, the FCC-ee would begin construction in 2028, and first physics would take place in 2039.

[...]

2.5 The Circular Electron Positron Collider

The Circular Electron Positron Collider (CEPC) is a proposal for a circular electron-positron collider

[...]

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 modern 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 by evolving directly into the final software, or 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. This results in different teams solving the same problems and implementing the same solutions for each subdetector.

An alternative to this is to develop a suite of tools or frameworks that are generic – capable of being used and deployed for a wide variety of different uses and detector types. In this way, we could greatly reduce the effort used to recreate the same solutions for each new detector, allowing more science to be done faster.

Online monitoring and data quality monitoring

The area of this that we have chosen to contribute to is the development of online monitoring and data quality monitoring tools. Data from testbeams is often not processed fully until well after it has been taken, sometimes after the testbeam has ended, due to contraints on time or processing power. If there were errors, incongruencies, or any other issues with the data, these issues cannot be identified immediately, and as a result may be present in multiple runs, spoiling data and wasting precious time during the already extremely time-sensitive environment of a testbeam.

Online monitoring addresses these issues by allowing the experimenters to see a "preview" of the data being collected. It can provide both a quantitative and qualitative look into how the detectors are responding, and what the data willl look like when properly processed, allowing any potential issues to be identified and fixed in a timely manner. This means that good online monitoring can improve the efficiency of a testbeam, increasing the "yield" of data from a given experiment.

Another aspect of this is data quality monitoring (DQM), which assesses the 'quality' of the data being taken. The definition of data's 'quality' will vary depending on the hardware, software, and goals of the experiment, but will usually constitute a some from of statistical measure. In this regard, data quality monitoring can be seen as an extension of the concept of online monitoring, focusing more closely on the quantitative aspects. In general, DQM provides the most benefit in more mature experiments, relying on previous experience with the detector and collected data to understand how the data appears when the device is functioning correctly.

In pursuit of all of the above aims, this chapter will discuss the Data Quality Monitoring

for High-Energy Physics tool (DQM4hep) developed for online monitoring and data quality monitoring, going into detail on its properties, principles, applications and development. Deployment and usage of the framework will be discussed in greater detail in Chapters 4 and 5.

The AIDA-2020 project

This work on DQM4hep takes place within the context of the AIDA-2020 project, 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 AIDA-2020 is to develop common tools and infrastructures for physics testbeams. The collaboration is split into work packages focusing on specific areas, and the work detailed in this thesis takes place within Work Package 5 for data acquisition systems for beam tests

The goal of this work package is to create a suite of tools that are designed with a variety of possible uses in mind, thereby reducing the work and development time necessary to implement data acquisition and monitoring setups, speeding up the planning and deployment of physics testbeams.

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, developed by Rémi Eté and Antoine Pingault. 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 framework is based upon 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 independent processes. The components that are specific to any particular use case are written by the 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,

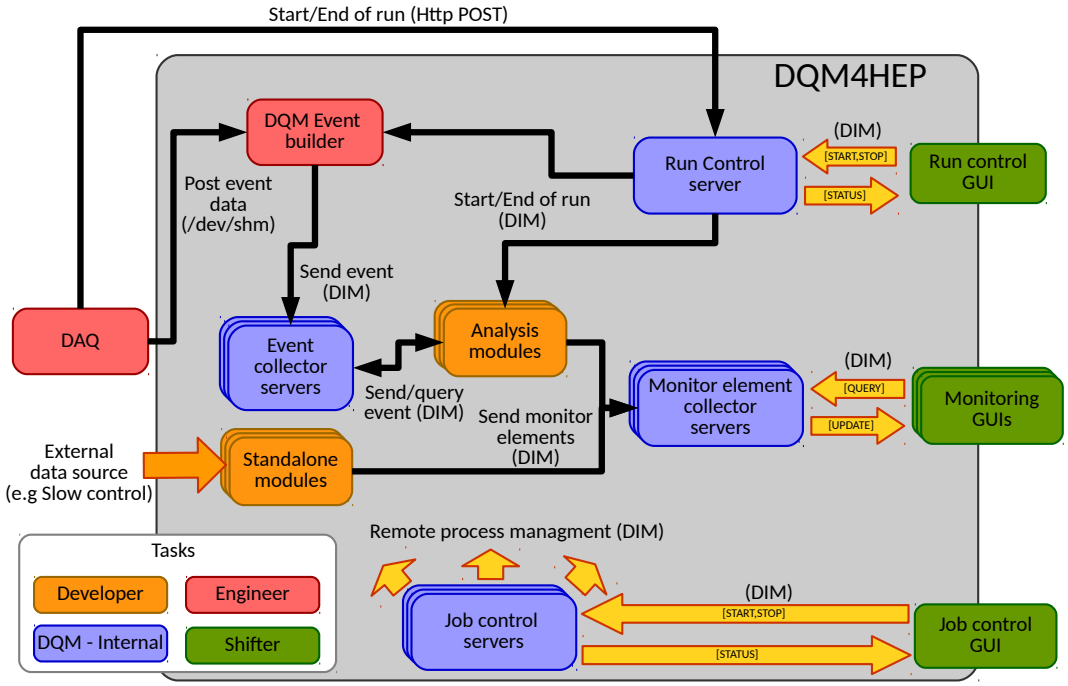

Figure 3.1: The global online architecture of DQM4hep. Each block is colour-coded to show which operator of the testbeam is responsible for the process.

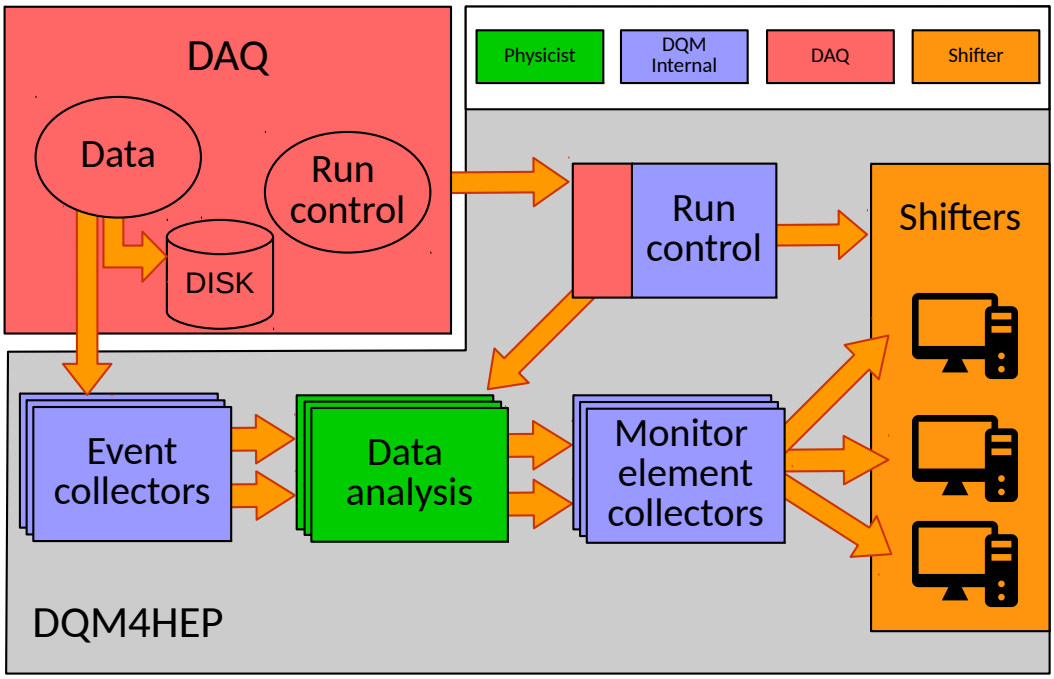

Figure 3.2: The structure of running DQM4hep online using an analysis module.

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.

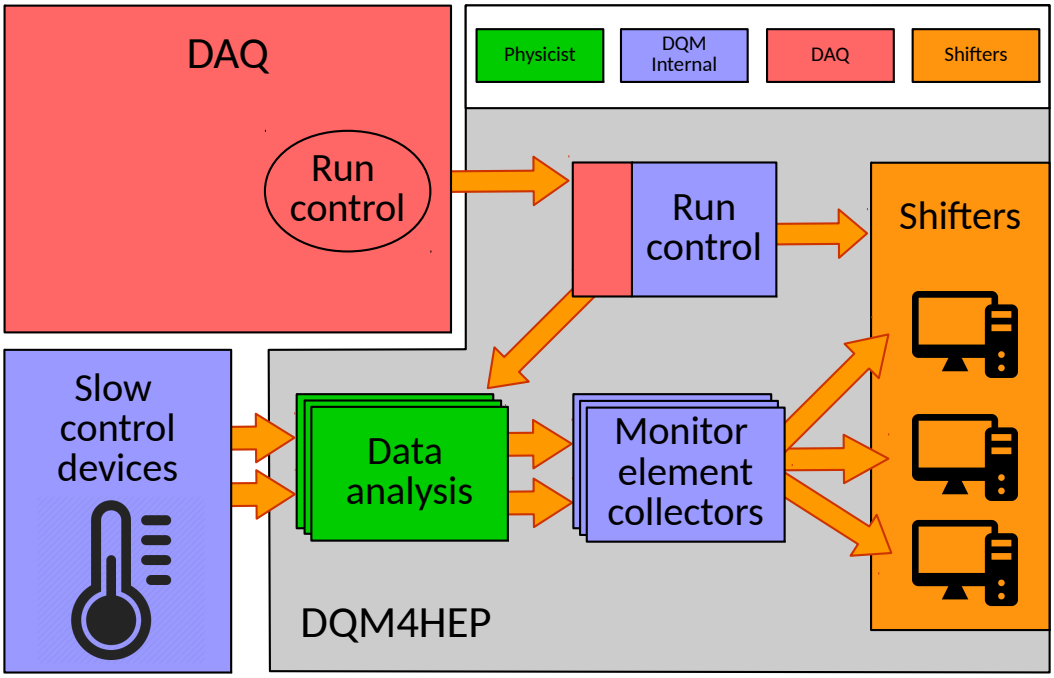

Figure 3.3: The structure of running DQM4hep online using a stand alone module.

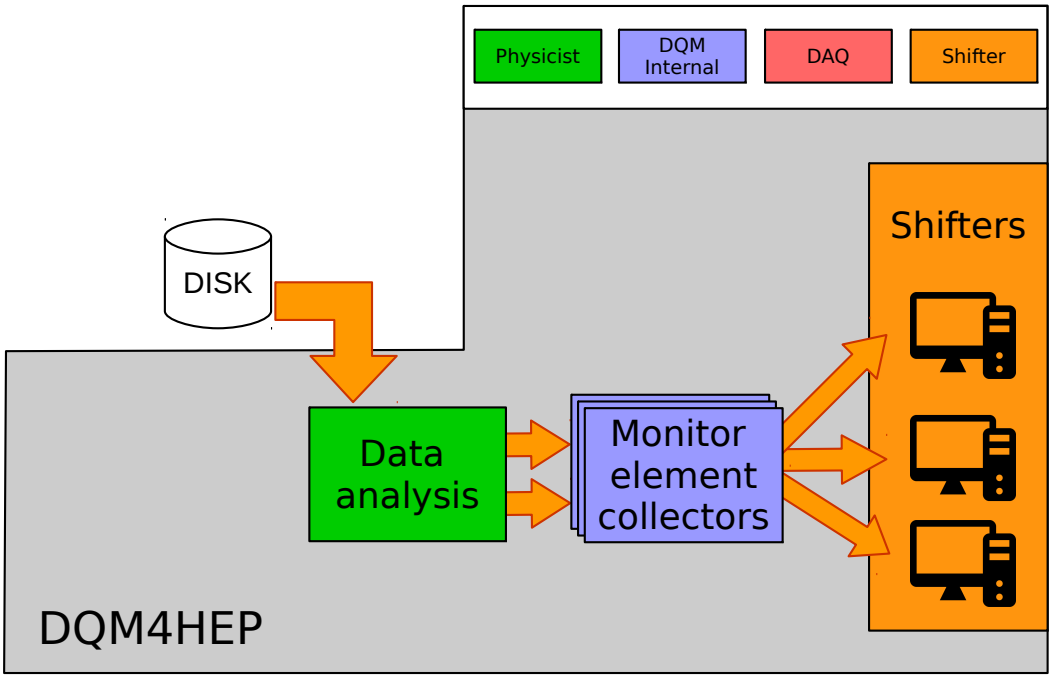

Figure 3.4: The structure of running DQM4hep offline using a file reader module.

3.1.2 Visualisation and graphical user interface

As of writing, the graphic user interface (GUI) and visualisation elements of the framework are still under active development for a new version. Therefore this topic will be split into two sections: one to describe the existing GUI, and one to discuss the motivations and goals for the new GUI under development.

Current GUI and visualisation

The current version of the GUI is built with Qt, a free and open-source toolkit and framework for creating OS-independent graphic user interfaces and widgets. The motivation for choosing Qt was that ROOT provides an option for integration between ROOT and Qt, allowing ROOT classes like `TCanvas` to be "embedded" into Qt widgets. This simplified the implementation of a GUI, allowing a graphical interface based on Qt to be written, then graphics from ROOT simply opened within the existing widgets and windows.

This interface is used in multiple places, including the run control process and the monitoring GUI. The monitoring GUI is built on a system of canvases. Each canvas can have multiple plots open, which can be resized, maximised, minimised, etc. and manipulated as normal for ROOT plots. The user can also create new canvases for more space to arrange plots.

In addition to this, there is an optional provision for a monitoring steering file, which contains presets of canvases, and the plots displayed on them. This is extremely useful when dealing with large datasets or large numbers of plots, as the plots required by the user can be opened automatically when the monitoring interface is run.

An example of the Qt-based monitoring GUI in use can 3be seen in Fig. 3.5.

New user interface and visualisation package

For the newer versions of DQM4hep, the decision was made to overhaul the GUI and visualisation packages, removing Qt from the framework and moving to a web-based interface.

The removal of Qt was motivated by two reasons. Firstly, the integration with ROOT provided some complications, since running DQM4hep's Qt-based GUI requires an installation of ROOT compiled with the `--enable-Qt` flag enabled. The majority of ROOT installations in remotely-accesible file systems based at CERN and DESY (which are heavily used for analysis and testbeams) were not compiled this way. Secondly, Qt was

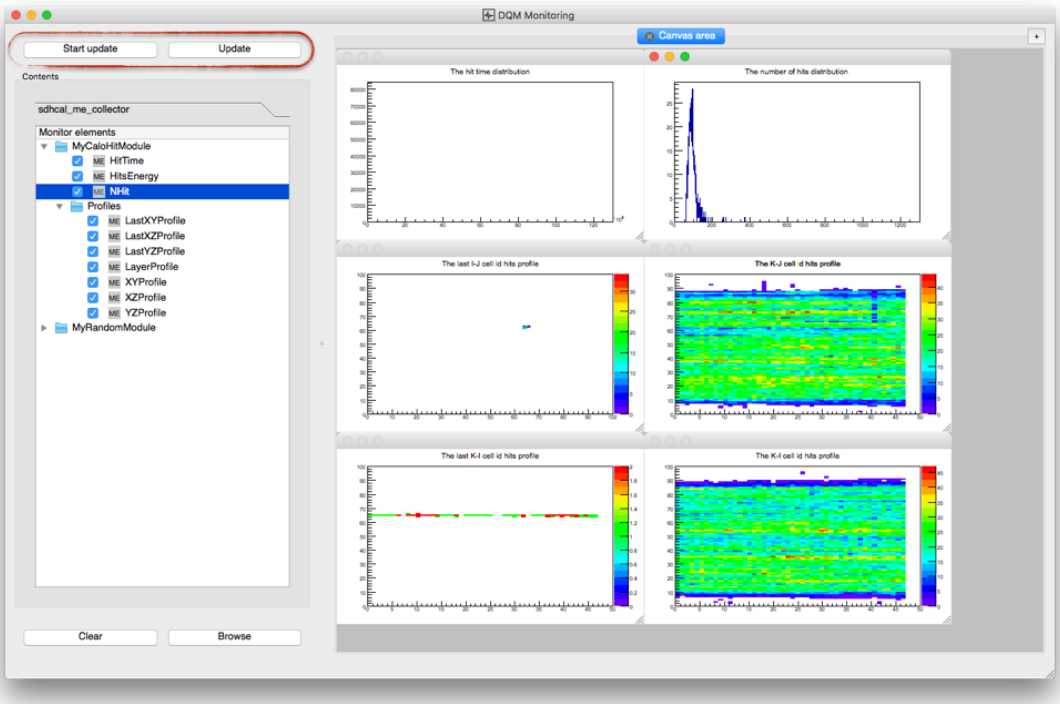

Figure 3.5: An example of the current Qt-based monitoring GUI in use.

an additional dependency that must be installed prior to use, making the software more dependent upon the operating system, compiler tools, and environment of the machine, and thus less generic and easy to use.

The removal of the Qt UI allows for greater freedom with UI development. The intended goal is to have a browser-based GUI, removing dependency on any external GUI libraries and allowing it to function on any device. This will also make it more user-friendly and convenient, as the interfaces for run control, networking, and data monitoring and quality display can be simply run in different tabs of a web browser.

As of writing, the web interface is under active development by Rémi Eté and is not yet complete. However, a mock-up of the web interface can be seen in Fig. 3.6.

3.2 Data quality monitoring

Data quality monitoring is a type of data monitoring where the data is tested using some form of statistical or mathematical process to produce a value corresponding to the "quality" of the dataset. This can take many forms, such as comparing an experimental dataset to reference data acquired from previous experiments, or requiring that the χ^2 or p-value of a dataset may need to pass a certain threshold to be considered valid.

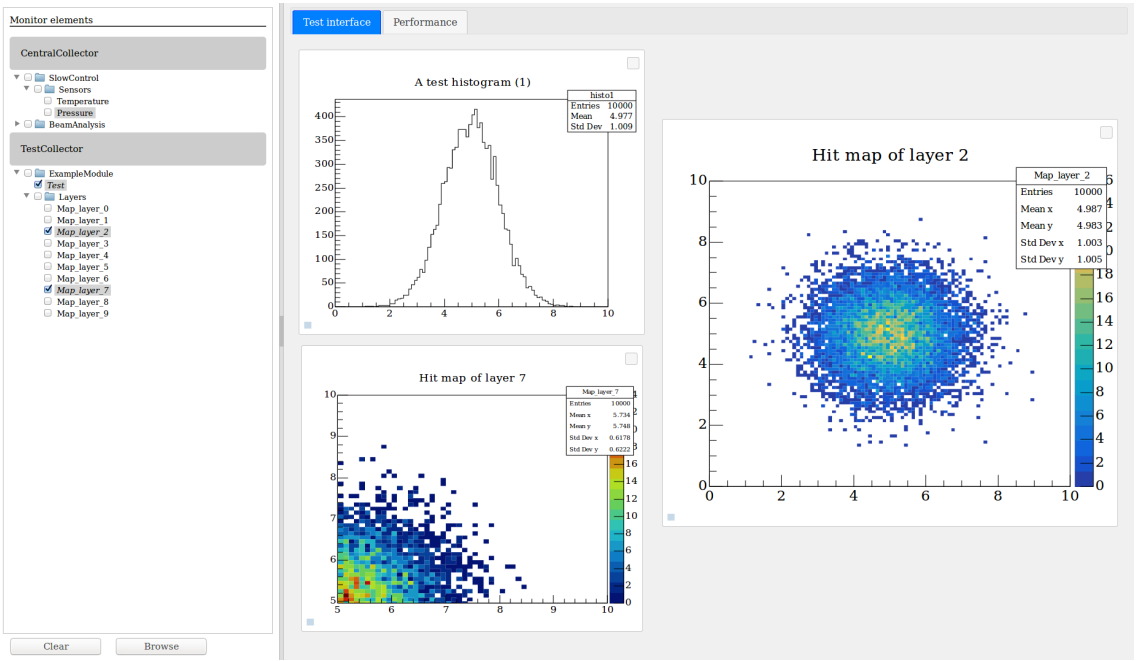

Figure 3.6: A preview of the planned web-based monitoring interface.

The definition of the "quality" statistic will differ according to a variety of factors such as the type of data, the aim of an experiment, etc. Common examples are p-values, or binary tests where data that passes had a quality of 1, and 0 otherwise.

One of the benefits of data quality monitoring is that it provides a more reproducible and robust set of checks on data-taking, allowing quantitative analysis of the performance of a detector prototype. It can also be used as a way for shifters without detailed knowledge of the hardware, software, or physics to determine whether the detector is performing as intended during a testbeam when experts are not available, by using the quality statistics as a guide.

Previous versions of DQM4hep did not have infrastructure to support data quality monitoring, but this was added during refactoring in preparation for the next release version. Once this was in place, this permitted an array of quality tests to be developed, implemented, and tested.

A quality test (or 'qtest') processes a series of monitor elements (ROOT TObjects) according to a set of criteria defined in the test's code. This test produces a numberical result between 0 and 1, referred to as the "quality". Within the framework, quality tests are self-contained C++ code files, which hook into the framework's system for execution. Quality tests are run by using the executable `dqm4hep-run-qtests` and a steering file to define paramters, such as which files to load, which quality tests to execute, and what the passing and failing boundaries are for each quality test.

The quality tests that have been implemented in DQM4hep are described in detail below. Each test requires a certain type of object as an input and has it's own definition of what the "quality" statistic represents. Some quality tests also require a reference to compare against the input data, which is also described.

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 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 exactly 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 functionality is also included for the sake of completeness.

3.2.5 Pearson 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. 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. 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.

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. The main action an analysis module must do is create a monitor element (a ROOT TObject) then emit it to the rest of the framework. Before this step an arbitrary amount of processing can be done, e.g. checking validation bits, thresholds, error-checking, and so on. Monitoring the data quality can be done from

within analysis modules but this is not recommended as dedicated quality tests (see section above) 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 for the promotion and uptake of a new framework is the lack of understanding or familiarity with it's use. Many research teams will continue to use 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.

The DQM4hep framework has two sets of documentation with different intended readers and different aims, so these will be discussed separately.

3.4.1 Doxygen documentation

Doxygen is a tool for automatically generating documentation resources for C++ code, relying on marked secctions of documentation written within the actual code itself. Doxygen is able to directly obtain the sturcture of code, objects, functions, etc. from the code, allowing it to automatically generate a complex and rich set of documentation that categorises and indexes objects based on their inheritance, namespace, etc.

Doxygen can also generate an HTML- or LaTeX-based document that can be used as a local reference guide or hosted online. This makes Doxygen a powerful tool for documenting the technical aspects of code, demonstrating hierarchies of functions and objects, and an extremely useful reference guide for large programs or frameworks.

While Doxygen documentation is extremely useful, it does have some limitations. Doxygen functions more as a technical reference for code, lacking any overviews or instructions due to it's automatic generation. This kind of documentation lacks a holistic element, and has no way for new users or those less familiar with the codebase to understand the overarching concepts. This can make it inaccessible for new users. The way this

was addressed will be dicussed in the next section.

DQM4hep has a Doxygen website hosted on the internet, which is available here [4].

3.4.2 User manual

The existing documentation featured the common elements of the framework – such as the plugin system, event interface, logger, and XML parser – explained in detail, with clear examples and straightforward advice on their use.

My contribution to this was to write in-depth explanations on the structure, usage, and creation of analysis modules and quality tests, the two parts that are most specific to end-users. The experience I acquired using the framework and deploying it on testbeams for the first time, as well as integrating it with different detectors, provided a strong knowledge base to write the user manual intended for a user approaching the framework for the first time. These testbeams are described in more detail in Chapters 4 and 5.

The user manual can be found online here [3].

Analysis module guide

There is an in-depth explanation of each of the component functions of an analysis module, discussing which actions or data processing should be done in each, and their intended purposes. There is also an explanation of the XML steering files that are necessary to run an analysis module, and instructions on how the executable is run, along with it's arguments.

In addition to this there is a worked example of an analysis module from start to finish. A simplified particle physics detector and it's data format is defined, and then the reader is lead through the process of writing an analysis module step-by-step and function-by-function to obtain certain plots and results from the data. This is intended to give a more concrete demonstration of usage, as an easier to follow example. This section is also written in simple and plain English in order to be as accessible as possible.

Along with this documentation there is the `dqm4hep-example` Github repository [cite], which contains all of the code used in the analysis module user manuals, as well as the required cmake and make files to begin compiling projects right away. The intention of this repository is for new users to make a copy, write their own modules, and have everything needed for compiling and running them "out of the box" straight away. The user manual references this repository, so that users can see the files from the worked example in their full form.

Quality testing guide

There is extensive documentation of quality testing; each quality test is described in detail, including their purpose, output, required parameters, and optional parameters. In addition to this, there is a guide for how to run quality tests, including an explanation of running from the command line and an in-depth look at the structure of the XML steering files required.

Finally there is a section explaining how to write new quality tests. This includes a detailed explanation of the purpose of each function within a quality test file, what the required outputs are and how to utilise them, error testing, and advice on maintaining a style consistent with the rest of the framework.

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 originally developed during testbeams of the Silicon Tungsten Electronic Calorimeter (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 nature.

In this chapter, deployment, testing and usage of DQM4hep on testbeams within the AIDA-2020 collaboration will be discussed in detail. For a testbeam where DQM4hep was deployed outside of the AIDA-2020 community, see Chapter 5. Three testbeams will be described in detail, as they represent the different stages of using DQM4hep as a data monitoring tool, from first implementation and deployment, to further developments, and finally as a mature tool integrated into the workflow of a testbeam.

CALICE testbeams

The CALICE testbeams were done with the *Forschung mit Lepton Collidern* (FLC)[1] group based at DESY in Hamburg, working on the Analogue Hadronic Calorimeter (AHCAL) prototype during its development. Regular testbeams were held at the DESY II

[1] EN: Research with Lepton Colliders

synchrotron at DESY in Hamburg, Germany and at the Super Proton Synchrotron (SPS) at CERN in Geneva, Switzerland. The goals for these testbeams varied over time but common focuses for the hardware were power-pulsing tests, and commissioning and calibration of new detector boards to test variations and changes to the manufacturing process. The testbeams were often used as testbeds for data acquisition electronics and software, such as the EUDAQ data acquisition software, the BIF device, and DQM4hep.

DQM4hep was used as an online monitoring and data quality monitoring tool for AHCAL tesbeams 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.

The AHCAL prototype

The Analogue Hadronic Calorimeter (AHCAL) is a sampling calorimeter formed of steel absorber plates and plastic scintillator tiles, read out by silicon photomultipliers (SiPMs) as active material [5]. One of the important features of the AHCAL is that the prototypes were designed to be made using techniques suitible for mass production, such as injection-moulding and automated foil-wrapping of the scintillator tiles, and pick-and-place assembly of the layers and their electronics. It also uses power pulsing – rapidly cycling power so that the electronics are active only when the beam is present, according to a known beam structure. This helps to reduce power consumption and heat production, making cooling the layers easier.

4.1 May 2016 at DESY II

The first deployment of DQM4hep on an AHCAL testbeam was at DESY II during May 2016. The testbeam was to be two weeks in duration, following a one-week setup and preparation period. Besides testing the deployment and usage of DQM4hep, the goals of this testbeam where to test MIP calibration of a new AHCAL base unit (HBU), to test the power pulsing feature, and to perform TDC calibrations. In addition to these goals for the AHCAL, a device called a Beam Interface (BIF), another part of the ongoing work of AIDA-2020 Work Package 5, was being tested.

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

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.

4.1.1 Data format

The data for the AHCAL is in the Linear Collider Input/Output (LCIO) format, using an object type called LCGenericObject, which is a generic format for use when the existing data formats are not suitable. It comprises two parts: the block of data itself, held in 14-bit numbers; and a header containing user-defined parameters, in this case a timestamp, a typename for the object, and a description of the data contained in the object.

The structure of a single event in LCGenericObject format can be seen below, which is the result of using the `dumpevent` tool to dump the contents of an LCIO event to the command line:

```
--------------- print out of LCGenericObject collection ---------------

 flag:  0x0
parameter DAQquality [int]: 1,
parameter DataDescription [string]: i:CycleNr:i:BunchXID;i:EvtNr;i:
    ChipID;i:NChannels:i:TDC14bit[NC];i:ADC14bit[NC],
parameter Timestamp [string]: Tue, 09 Feb 2016 18:20:43 +0100,
parameter TypeName [string]: CaliceObject,

[   id   ] i:Type,i:EventCnt,i:TS_Low,i:TS_High - isFixedSize: false
-------------------------------------------------------
[00000004]  i:0; i:15; i:15; i:0; i:36; i:12423; i:12422; i:12421;
    i:12420; i:12419; i:12418; i:12417; i:12416; i:12415; i:12414;
    i:12413; i:12412; i:12411; i:12410; i:12409; i:12408; i:12407;
    i:12406; i:12405; i:12404; i:12403; i:12402; i:12401; i:12400;
    i:12399; i:12398; i:12397; i:12396; i:12395; i:12394; i:12393;
    i:12392; i:12391; i:12390; i:12389; i:12388; i:12459; i:12458;
    i:12457; i:12456; i:12455; i:12454; i:12453; i:12452; i:12451;
    i:12450; i:12449; i:12448; i:12447; i:12446; i:12445; i:12444;
    i:12443; i:12442; i:12441; i:12440; i:12439; i:12438; i:12437;
    i:12436; i:12435; i:12434; i:12433; i:12432; i:12431; i:12430;
    i:12429; i:12428; i:12427; i:12426; i:12425; i:12424;
```

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`.

The `ADC14bit` and `TDC14bit` arrays contain binary data, which is represented above converted directly to decimal. However some bits represent data other than the actual ADC or TDC, such as validation bits, hit bits, etc. An explanation of the structure of these bits can be seen in [ref].

[...]

4.1.2 Results

Over the course of the preparation week, the foundations were laid for the analysis module. This involved gaining a familiarity with the data structure, loading data from previous testbeams into DQM4hep offline, and attempting to read it in basic ways. The first module was not ready for the beginning of the testbeam proper, but a few days afterwards we were able to produce plots in DQM4hep from "nearly-online" testbeam data.

The first analysis module developed was the `AHCALRawModule`. The majority of the processing in this module was decoding of the data from the binary format and extracting the information from it. After this, validation bits and hit bits in the data were checked to classify data as 'good' or 'bad' hits. Then the actual ADCs and TDCs were filled into their respective histograms.

The first module acted as a proof-of-concept, and once this was done further work started on creating more modules with a wider variety of features and plots to provide better coverage for online monitoring. We created and refined two separate modules during this testbeam – `AHCALRawModuleChannel` and `AHCALRawModuleGlobal`.

The channel module created a per-spectrum channel of all ADCs, integrated over the whole run. It was able to load a number of individual channels, though due to the memory requirement, it could not track all channels simultaneously. To work around this, we implemented a facility for the module to read which channels to monitor from the XML steering file, so that these could be defined at runtime. An example of some of the per-channel spectra created using this module can be seen in Fig. 4.1

The global module produced a 2D histogram containing cells for each channel, coloured

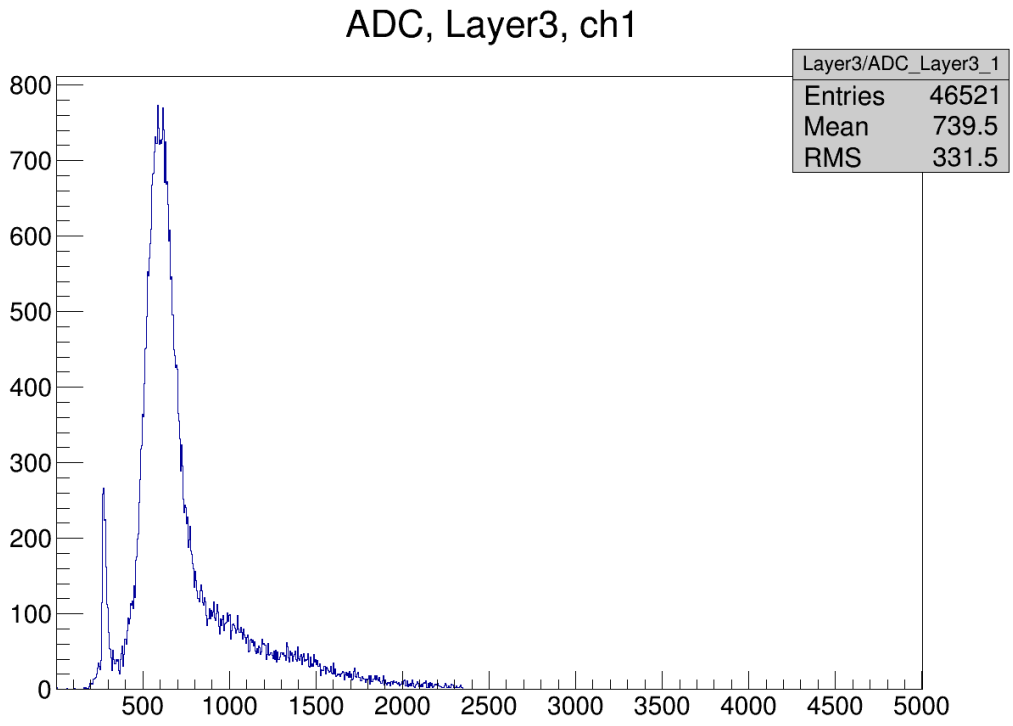

Figure 4.1: Histogram produced by `AHCALRawModuleChannel` showing an ADC spectrum of a single channel for one run. The pedestal can be seen at approximately 300 and the MIP peak at around 650.

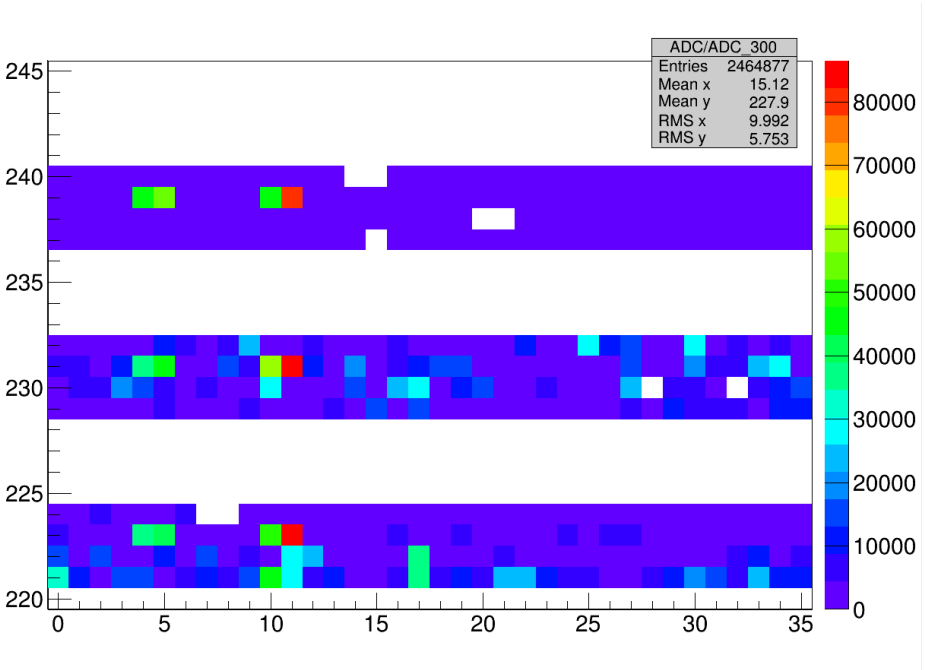

Figure 4.2: Histogram produced by `AHCALRawModuleGlobal` showing all ADCs exceeding 300 over a single run. Dead or nonresponsive channels are seen as white squares. The horizontal gaps are due to the fact that some ChipIDs were not present.

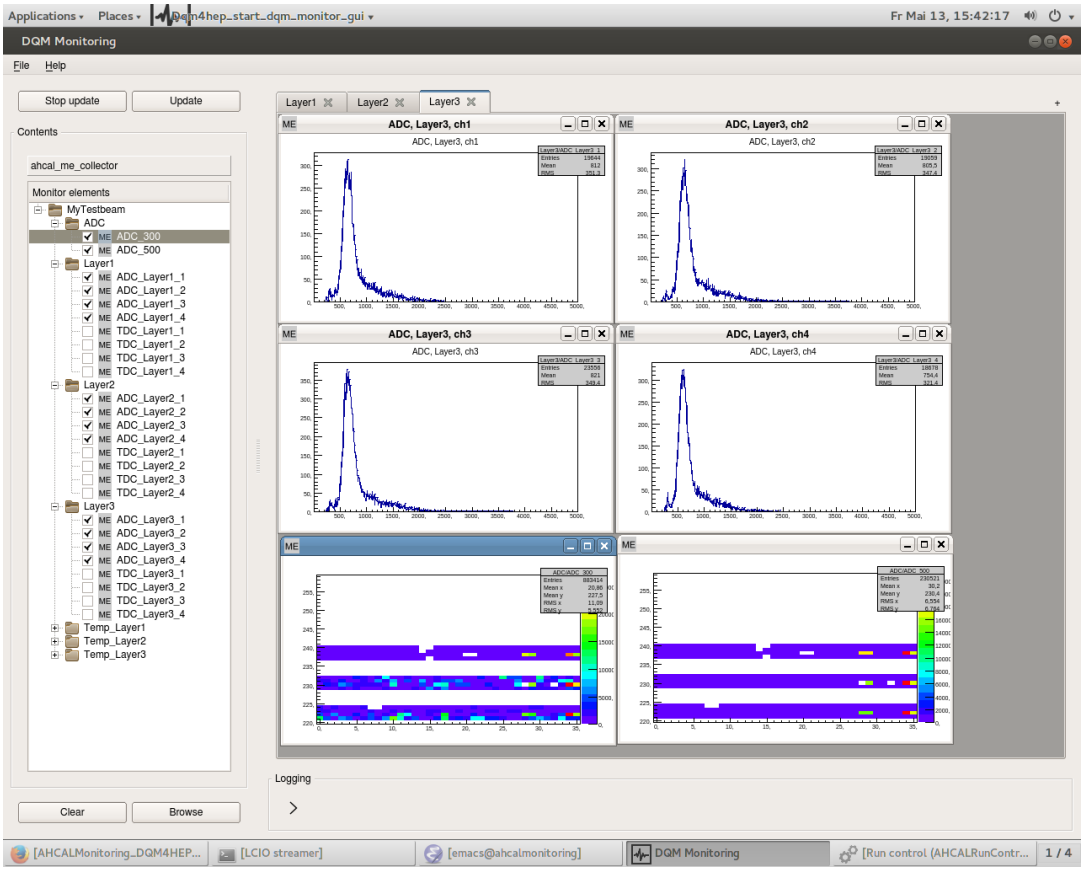

Figure 4.3: A full screencapture from the May 2016 testbeam showing the monitoring interface in use during power pulsing tests, displaying a MIP scan.

for the ADC in that channel. This didn't produce a hitmap as the geometry information was not available in this plot, but did allow easy identification of dead channels and channels that were in the beamspot. An example of this can be seen in Fig. 4.2.

Overall, once the monitoring had been set up and initial bugs and problems fixed, it became a routine tool of the testbeam. This was made easier by the usage of XML steering files for the monitoring interface canvases, allowing groups of plots and histograms to be automatically opened when the monitoring interface was started. This meant that with little effort, all information necessary for monitoring was easily available.

An example of the monitoring interface in use can be seen in Fig. 4.3.

4.2 July 2016 DESY II

The next usage of DQM4hep was a testbeam during July 2016, also at the DESY II facility. The testbeam was to be one week long, with a one-week preparation period. The primary goal for DQM4hep in this testbeam was to establish hitmaps of the calori-

meter. This would give a visual representation of the layers, allowing identification of the beamspot, allowing dead or mis-calibrated channels to be identified visually.

Creating a hitmap for the AHCAL was nontrivial, as the information coming from the data acquisition device and stored in LCIO format did not encode location. Each channel was instead identified by its "electronics number" – a combination of the ChipID of the board the channel was located on, and the number of the channel on that board.

Each layer was formed of four boards (each one with a ChipID), each of which contained sixteen layers. The orientation of the boards, which boards were in a layer, and the order of the layers in the stack were all changeable, so an additional requirement for the hitmap function was that it could take an external geometry file that could be changed or automatically generated.

DQM4hep has internal functions for parsing XML data, so an XML file was chosen as the format to store the geometry data. By making this an XML file, it avoided hard-coding the geometry into the framework itself, allowing the geometry data to be changed at runtime. XML also has the benefit of being human-readable. The AHCAL team already used an internal format for the geometry file for offline processing after testbeams, which was constructed and converted into different fromats via a shell script, so this could be replicated for the DQM4hep XML file.

For analysis modules needing geometry, the XML file was given as a required parameter for the steering file, which then built a C++ map of the correspondence between electronics number and (i, j, k) co-ordinates of each channel in memory during initialisation. Then a function called `electronicsToIJK` was written that took the electronics number as the argument and returned the position of the channel in geometric co-ordinates. A further function was written, called `IJKToElectronics`, that performed the opposite operation, but this was not used.

Using these new functions, another analysis module called `AHCALHitmap` was written that created a two dimensional histogram, with each bin representing a channel on the x and y axes for a single layer. This histogram was then filled with with the ADCs of that channel for the whole event, producing a hitmap.

The analysis modules for creating hitmaps were prepared ahead of the testbeam, and used extensively. They were used to identify channels that were dead or nonresponsive, as well as to confirm already-known nonresponsive channels.

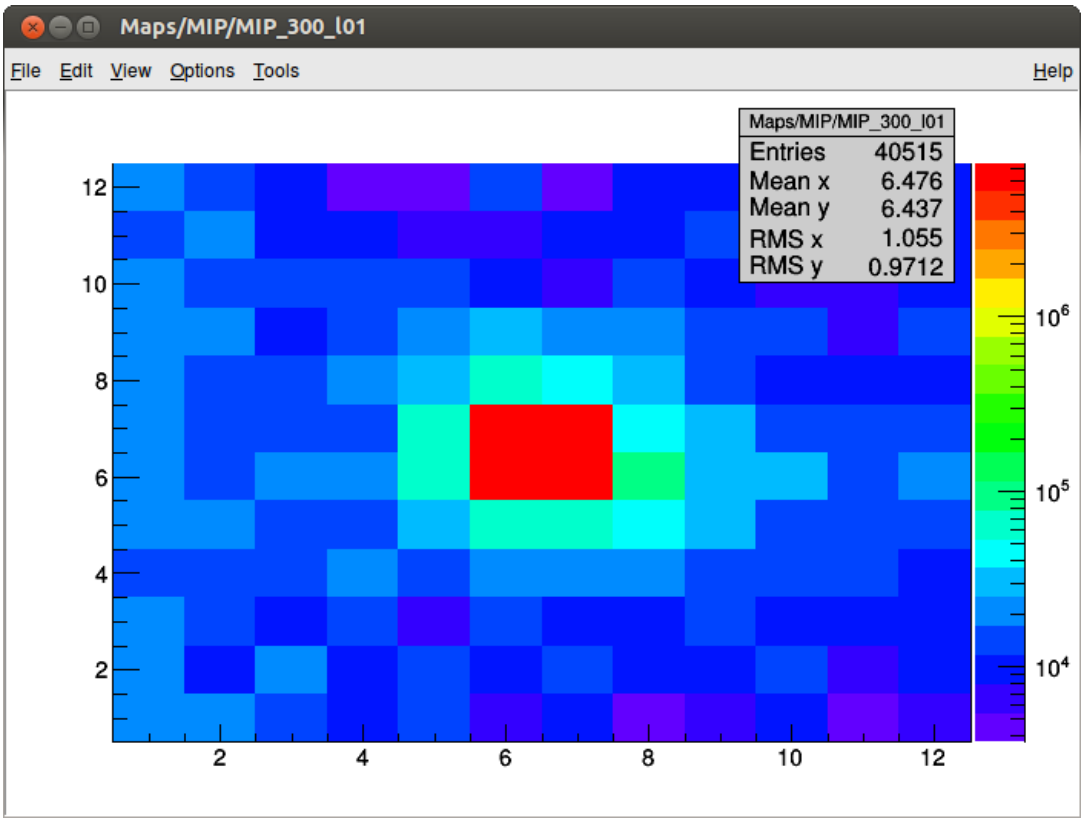

Figure 4.4: A hitmap for a single layer during the July 2016 testbeam, showing all channels with ADCs higher than 300. The beamspot is clearly visible in the centre.

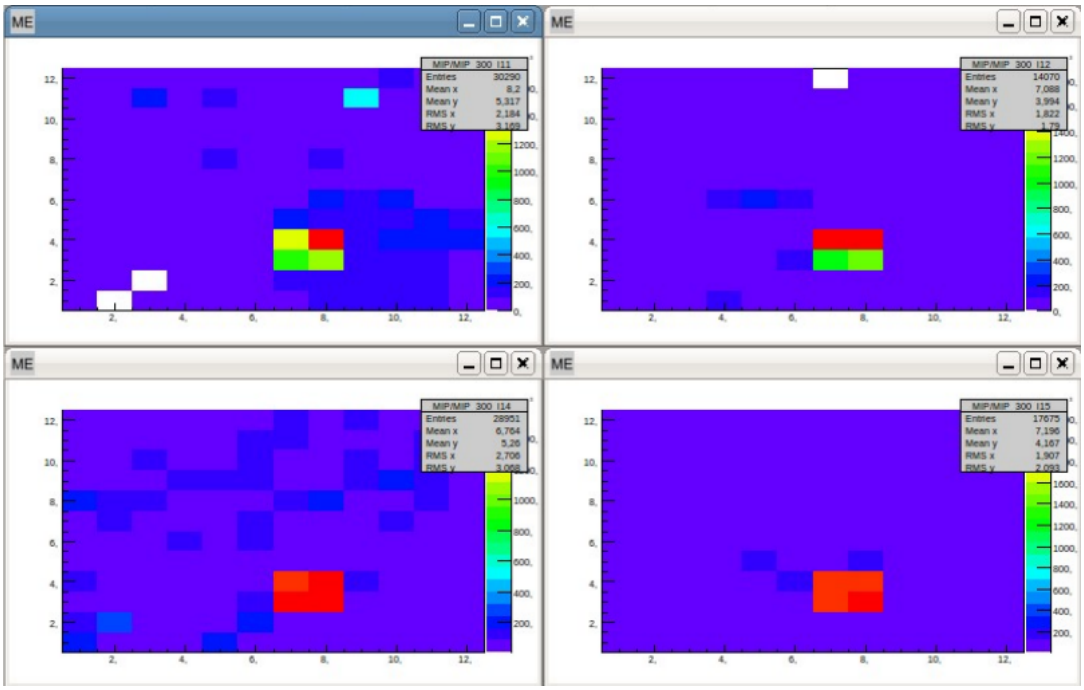

Figure 4.5: A collection of hitmaps for four layers in the stack during the July 2016 testbeam. The beamspot is visible, as are several dead or unresponsive channels in the top-left and top-right hitmaps.

4.3 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. One of the goals for this testbeam was to evaluate the performance of the power pulsing feature in magnetic fields up to 1.5T. The process of manufacturing the detector layers and boards was being automated, and a larger number of layers were available for this testbeam, so it also presented a way to test using a larger number of channels than before.

Part of the programme for the testbeam was to use the electron and muon beams available at the SPS for calibrating the newly-produced layers, as well as doing an energy scan with a pion beam.

The monitoring with DQM4hep in this testbeam included analysis modules that monitored the individual channels of all 40 large layers, as well as producing hitmaps of several types, such as unweighted, ADC-weighted, and near-pedestal. Standalone modules monitoring the temperature of the detector hardware was also used.

At this point in the testbeam process, the online monitoring system with DQM4hep had matured, partially due to the data format of the detector having been fixed for some time. Experience with running, using, and modifying the analysis modules had been disseminated throughout the team, and team members wrote their own analysis modules for producing plots.

Because of this, during the testbeam DQM4hep was used as intended – as a tool for shifters to use to diagnose and troubleshoot problems with the beams or detectors. A specific expert on DQM4hep was not necessary, as enough people operating the testbeam understood DQM4hep to be able to modify analysis modules on the fly according to their needs.

[Photographs from the installation and testbeam area go here]

[Plots from the testbeam]

Chapter 5

IDEA testbeams

[new quote split
over two lines]

[author]

While DQM4hep was intended from the beginning as a generic tool, it was developed largely within the AIDA-2020 collaboration, which also promoted a variety of standards and guidelines for data acquisition devices, data formats, etc. In addition to this, it was also only tested on calorimeter-type detectors within the CALICE collaboration. To be sure that DQM4hep was truly generic – capable of adapting to *any* detector – it was necessary to test it on a wider variety of detectors outside of the AIDA-2020 and CALICE collaborations.

An ideal opportunity arose for this in the form of the IDEA testbeam. This testbeam was run by a collaboration of Italian and US universities working on four different particle physics detectors of different types, intending to run a single combined testbeam of all four instruments at the CERN SPS in September 2018. DQM4hep was offered as a possible unified monitoring solution that could integrate information from all four detectors into the same tool. This provided convenience for the teams operating the testbeam and detectors, and an extremely valuable opportunity to test DQM4hep out of its established operating range.

5.1 Introduction

The combined testbeam took place between in September 2018 at the CERN SPS beamline facility. The testbeam itself lasted a week, from the 5th-12th September, with a preparation period of one week before this for installation, calibration, etc. [...]

Detectors 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 Dual REAdout Method (DREAM) calorimeter, also called the RD52 calorimeter, is a dual-readout calorimeter built, developed and tested by the RD52 collaboration by researchers based in universities at Gagliari, Cosenza, Pavia, Pisa, Rome, Iowa State and Texas Tech.

The aim of the calorimeter is to use both the Čerenkov and scintillator techniques simultaneously (hence "dual-readout") to improve calorimetry, especially using calorimeters to measure four-vectors of jets and single hadrons, which suffer a reduced precision compared to electrons and photons. By comparing signals from Čerenkov light and scintillator light, the electromagnetic shower fraction can be measured on an event-by-event basis, eliminating the effects of fluctuations.

Muon chamber and preshower

[...]

Silicon photomultiplier GEM

[...]

Drift chamber

[...]

5.2 Monitoring

[...]

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 [...]. 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

Writing file readers for the different detectors meant first understanding the structure of their data and file types. Once data has arrived from the data acquisition device, each of the detectors wrote to a different 'raw' format. However, the RD52 calorimeter, drift chamber, and muon and preshower all produced ROOT ntuple files as part of their data acquisition process, in addition to different file formats. It was decided to use these ROOT ntuples as the file format to read into DQM4hep, as due to support for ROOT within the framework, reading data from ROOT ntuples is simpler.

For each of these three instruments a file reader was developed that walked through the ROOT trees, extracting data from the leaves event-by-event, then converted it to DQM4hep's inbuilt `GenericEvent` format. Choosing to make them into GenericEvents meant that there was no need to implement an additional event type, simplifying the connection between the file readers and analysis modules. The GenericEvent type was also easily suited to the type of data, as the majority of the data were series of numbers, which were easily converted into C++ vectors to store in the GenericEvent.

An additional motivation for using the ROOT ntuple as the data source was that some of the file type, notably the drift chamber, had not finalised their data structure at the beginning of the testbeam. Using the higher-level structure provided by a ROOT file would be easier to change in the future than reading in data in a binary, hex, text or CSV format.

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. It also allowed the parameters in the XML header, which included run numbers and physical information of the detector, to be passed into DQM4hep and the analysis module, using the `core::Run` object type.

5.2.2 Analysis modules

During the course of the testbeam, four analysis modules were developed, one for each detector. To begin with, these were 'dummy' modules – analysis modules that receive

events then do nothing. Dummy modules like this are required to run file readers offline, but are simple to create as they can be produced from a template with minimal changes.

Once the file reader for the RD52 calorimeter was complete, the dummy analysis module for this detector was then changed into a functional module, `RD52MainModule`, to produce plots from the data. The RD52 was chosen as it was the most data-rich of the detectors in the testbeam, and also contained the ancillary detectors, which allow particle selection efficiencies to be studied.

During the course of the testbeam, the `RD52MainModule` was developed to read in and arrange data from the ROOT ntuples and arrange them into distinct vectors for the ADCs, TDCs, pedestals, and ancillaries. Following this, pedestal subtraction of all ADC and TDC channels was completed. Then the malfunctioning tiles and electronics that meant data had to be re-routed through other available channels was integrated so that the ADC information in the analysis module represented the full state of the detector was completed.

There was insufficient time at the testbeam to develop the monitoring more complex than this, but further development continued offline with saved data from the testbeam to continue the proof-of-concept. This will be discussed in more detail in the following section.

5.3 Results

Monitoring within DQM4hep was able to replicate all the existing monitoring solutions, combining the monitoring tools for all four detector systems into one framework. [...] However due to time constraints, more complex monitoring such as online processing was not implemented during the testbeam. [...]

Following the testbeam, it was decided that DQM4hep could be used to perform some offline analysis functions in addition to the work at the testbeam, as a further proof of its ability to do this at testbeams. [...]

Several analysis goals were outlined to attempt to implement in DQM4hep. These were performing the tower ADC calibration, the ADC to energy calibration, and particle selection efficiencies.

5.3.1 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.

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.

5.3.2 ADC to energy calibration

[...] It was known from prior testbeams that the response of the calorimeter was linear with ADC, therefore the energy calibration could be extrapolated from a single datapoint. [...]

5.3.3 Particle selection efficiencies

An important result for the calorimeter was determining the efficiency with which it could determine the beam composition and select for certain types of particles based on kinematic properties. The 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 the calorimeter itself to

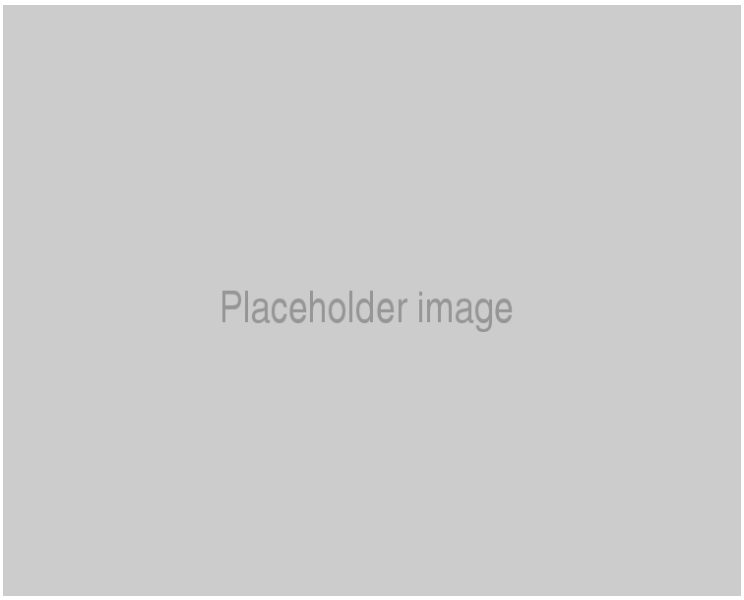

Figure 5.1: The plot of ADCs for each tower before calibration.

perform a second particle selection, and then compare the two.

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

[...]

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} 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

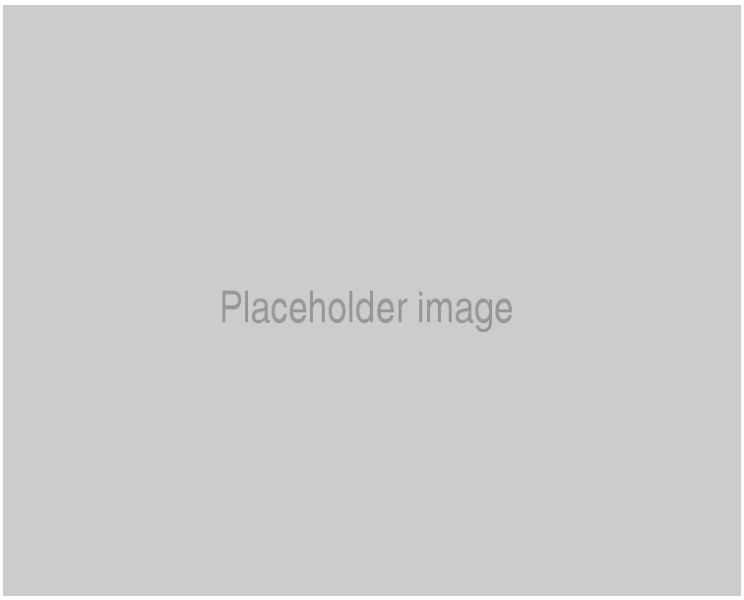

Figure 5.2: Comparison of the calibration plots for the ADCs before and after calibration.

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 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:

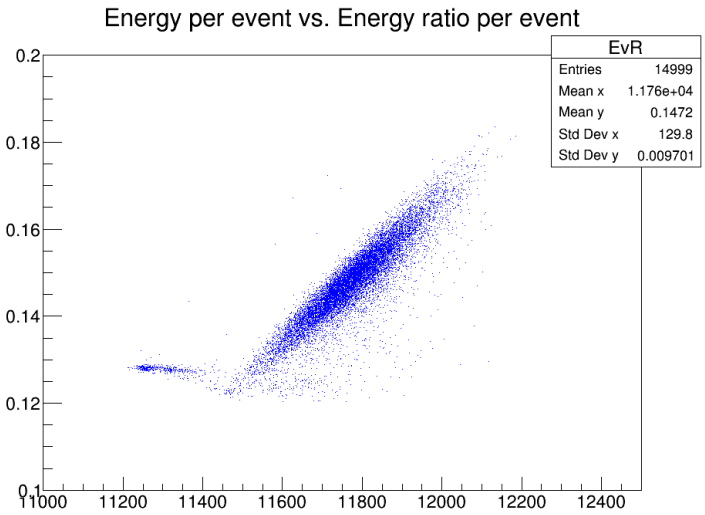

Figure 5.3: Plot of E against R for the secondary hadron beam with $n = 10$ (Run 12709).

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

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

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 The Standard Model (SM) 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 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.

[...]

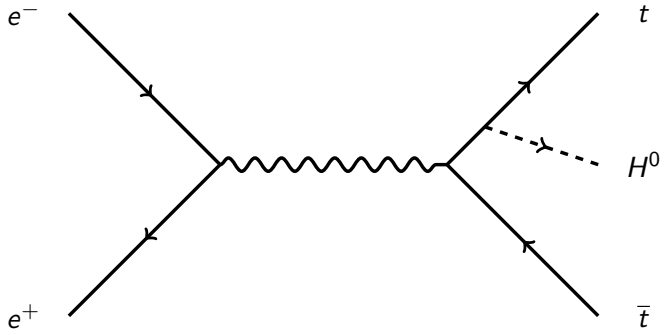

Figure 6.1: A Feynman diagram of the tth event.

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

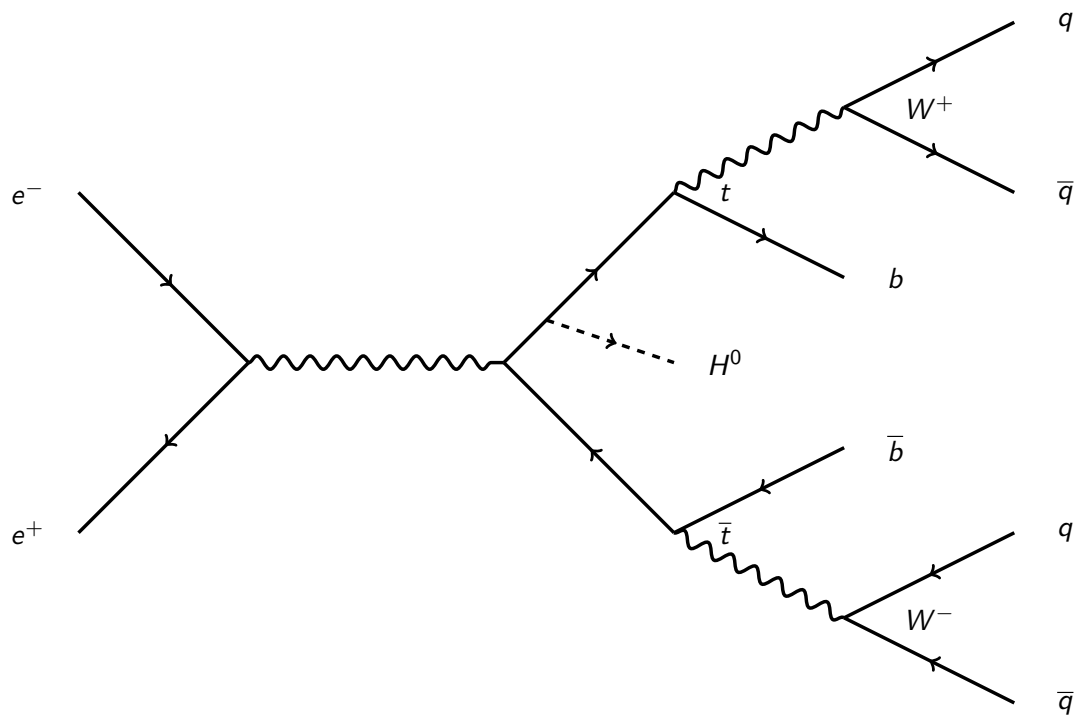

Figure 6.2: An extended Feynman diagram of the tth event, showing the fully-hadronic decay channel with the final state $q\overline{q}q\overline{q}b\overline{b}$, where q and $\overline{q}$ indicate a quark-antiquark pair.

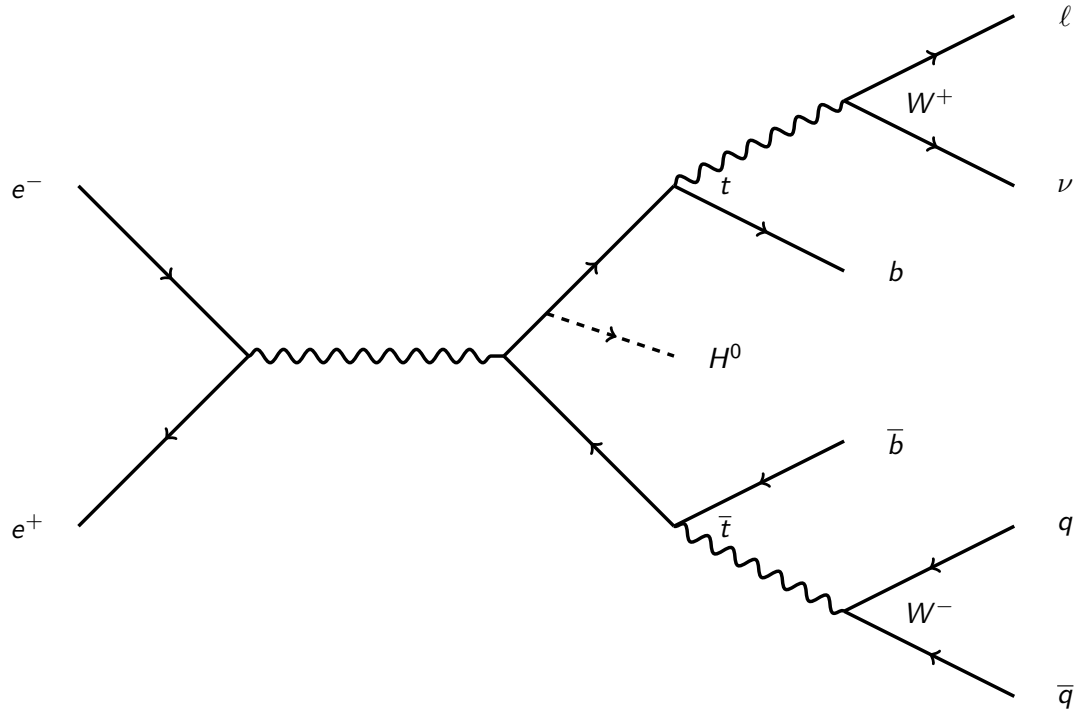

Figure 6.3: An extended Feynman diagram of the tth event, showing the semi-hadronic decay channel with the final state $\ell\nu q\overline{q}b\overline{b}$, where ℓ and ν indicate a lepton-neutrino pair of the same flavour but opposite [sign?].

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]

[...]

[...]

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:

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

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

[...]

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. 6.4). 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.

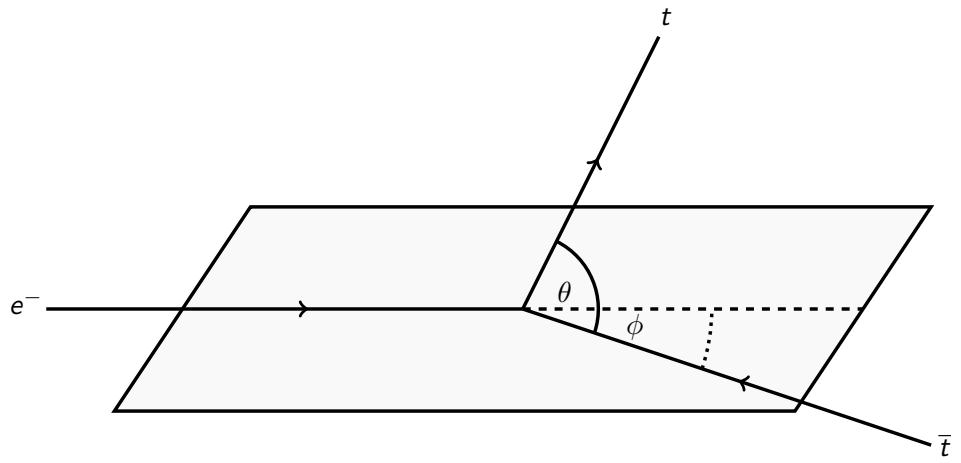

Figure 6.4: Geometric diagram of the up-down asymmetry in tth events. The paths of the electron and antitop quark, and the angle ϕ between them define a plane. "Up-going" top quarks go above the plane, "down-going" top quarks go below.

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 (PandoraPFAs) 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. 6

[2] R Ete, T Coates, and A Pingault. Dqm4hep: a generic data quality monitoring framework for hep. Technical report, 2018. 12

[3] Dqm4hep user manual. https://dqm4hep.readthedocs.io/en/latest/. Accessed: 2019-02-07. 23

[4] Dqm4hep doxygen pages. https://dqm4hep.github.io/dqm4hep-doxygen/doxygen/dqm4hep/master/index.html. Accessed: 2019-02-07. 23

[5] Felix Sefkow and Frank Simon. arxiv: A highly granular sipm-on-tile calorimeter prototype. Technical report, 2018. 26

Acronyms

ADC Analogue to Digital Conversion. 30

AHCAL Analogue Hadronic Calorimeter. 25, 26

ATLAS A Toroidal LHC Apparatus. 4

BIF Beam Interface. 26

BSM Beyond the Standard Model. 2, 4, 43

CALICE Calorimeter for Linear Collider Experiment. 25

CEPC Circular Electron Positron Collider. 3, 8

CLIC Compact Linear Collider. 2

CP Charge Parity. 43

CSV Comma-Separated Values. 36

DAQ Data Acquisition. 10

DESY *Deutches Elektronen-Synchrotron*. 26

DQM Data Quality Monitoring. 11

DQM4hep Data Quality Monitoring for High-Energy Physics. 11, 12

DREAM Dual REAdout Method. 35

EUDAQ EU [?] Data Acquisition. 26

FCC Future Circular Collider. 3, 8

FLC *Forschung mit Lepton Collidern*. 25

GUI Graphical User Interface. 17

HL-LHC High Luminosity Large Hadron Collider. 2

ILC International Linear Collider. 2, 5

ILD International Linear Detector. 6

JINR Joint Institute for Nuclear Research. 6

LCIO Linear Collider Input/Output. 27

LEP Large Electron Positron Collider. 8

LHC Large Hadron Collider. 2–4

MIP Minimum Ionising Particle. 26

PandoraPFAs Pandora Particle Flow Algorithms. 49

PFO Particle Flow Object. 47

QCD Quantum Chromodynamics. 4

SiD Silicon Collider. 6

SiPM Silicon Photomultiplier. 26

SiWECAL Silicon Tungsten Electronic Calorimeter. 25

SM The Standard Model. 2, 5, 43

SPS Super Proton Synchrotron. 10, 26, 34

TDC Time to Digital Conversion. 26

XML Extensible Markup Language. 36