A New Object-Oriented Physics Analysis Framework for the H1 Experiment

Andreas B. Meyer* **on behalf of the H1 Collaboration**

Universität Hamburg, Luruper Chaussee 149, 22761 Hamburg
E-mail: `Andreas.Meyer@desy.de`

Abstract: The H1 Collaboration is currently upgrading its analysis software. The new integrated object-oriented physics analysis environment is based on the RooT framework and comprises a new data storage scheme, a data access front-end, and a new event display. The analysis data is stored in three layers of separate files. The uppermost event-summary layer ('HAT') facilitates a fast selection via event classification and kinematics. At the particle level (second layer, 'μODS') a custom-built pointer relation ('H1Pointer') allows for direct access from the particle 4-vectors to the corresponding track and cluster information which is stored at reconstruction level ('ODS'). The user access to the data files is encapsulated using extensions to the RooT tree class in connection with a MySQL database.

1. Introduction

Similar to most large high energy physics experiments, in the 1990s, also the H1 Experiment made use of FORTRAN for most of the off-line software, i.e. the central reconstruction, the simulation software as well as the physics analysis software. At H1 the reconstruction and simulation output is stored in the BOS format [1]. The DST (Data Summary Tapes, 15 kB/event), comprising reconstructed tracks and clusters, as well as other detector information essential for physics analyses, are held on disk. The first step of a physics analysis is then to select the events of interest and to copy the relevant subset of information onto so-called N-tuples. In the past five years the majority of N-tuple analyses was done with the Fortran based packages HBOOK [3] and PAW [4] developed at CERN.

One of the features of the new RooT-package [5], in contrast to PAW, is the concern for the homogeneity of both analysis code and data. RooT makes use of C++ for its own implementation as well as the analysis code. A C++ interpreter [6] is integral part of the RooT package. This allows to use C++ as one common programming language and coding

*Speaker.

convention for both analysis and physics classification code. C++ supports object-oriented programming techniques as a powerful means to achieve the required maintainability and extendibility of the software. The RooT package not only replaces the full set of capabilities of PAW, it also optimizes the persistent storage of data. The graphical user interface of RooT allows to directly produce the representation of the RooT data e.g. in histograms. For these and other advantages of RooT, such as its already widespread use by the HEP community, the H1 Collaboration has chosen RooT as the basis for its physics analysis software upgrade project.

The H1 experiment is one of the four big experiments at the electron proton collider HERA. It investigates the collisions of electrons or positrons with protons at a center of mass energy of 320 GeV. A detailed description of the experiment can be found in [7]. The H1 detector and reconstruction software have been running stably and successfully for many years. The scope of the software project was therefore restricted to the reorganization of the quickly evolving parts of the software, namely the physics analysis code. The reconstruction and simulation software were left untouched. The new framework facilitates a transparent exchange of algorithms between different working groups (horizontal portability) as well as the re-use of user analysis code during the central production of H1 physics data sets (vertical portability).

Specifically the goals of the upgrade software are to incorporate and support all H1 physics analyses in one unique and common framework and to standardize the physics algorithms, such as kinematic reconstruction, calibration selection criteria and particle identifications. The persistent storage of readily accessible analysis results streamlines the exchange of information between different analysis groups, and thus makes expert knowledge reusable by non-experts. The thresholds of starting a new analysis is effectively lowered by a faster, more selective, direct and therefore efficient access to the data. It is expected that the experience with the introduction of a modern programming language along with the paradigm of object oriented design and implementation of the code will provide physicists and students with profitable experience for future projects.

Data Storage Model

In the new scheme, the data is stored in three different layers.

ODS: Reconstructed tracks and clusters as well as important detector information is stored in the so-called ODS (Object Data Store). The contents of the ODS corresponds to the former DST in a 1-to-1 fashion, this way facilitating full backward compatibility. The ODS format is re-convertible into DST. This allows for a smooth migration of the existing physics analysis software, which uses DST.

μODS: The μODS, see fig. 1, contains a list of particle candidates, i.e. four-vector objects, which are combined from track and/or cluster information. Each of the particles stores a pointer relation to the ODS-tracks and clusters that it originates from. In addition, to allow direct access to specific identified particles, e.g. the ep-scattered outgoing electron, muons, pions, kaons, combined hadronic final state particles, a list of pointers is created for each particle type. Composed particles, such as J/ψ, D^*, Jets etc. are retrievable through lists of pointers to the corresponding daughter particles. The different pointer list objects hold accessor functions that obey a common interface, but are specific to the particle type. The pointer relations allow quick direct navigation to the requested information. The use of loops and if-statements is minimized. Finally, for particles of particular interest, the μODS also contains a minimal amount of reconstruction information, e.g. for the outgoing electron the full track and cluster information is stored at the μODS. With the information provided, the μODS is largely sufficient for most analysis purposes, and still only comprises about 1kB/event on average. The export of selected μODS events by individual users to off-site computers is thus straightforward.

HAT: The HAT ('H1 Analysis Tag') contains event summary level information, such as event kinematics and particle multiplicities as well as specific analysis tags. With this information it is possible to select specific events solely on HAT information providing a functionality much superior to commonly used index files (event directories). At present the size is 400 B/event, i.e. a factor of 40 smaller than the former DST or the ODS. It is mainly this feature that increases the speed of the selection considerably. At present the size of both HAT and μODS is being reduced, mainly by means of removal of redundant and/or unused information. A central selection facility that uses HAT level information is implemented in the H1Tree class which is explained below.

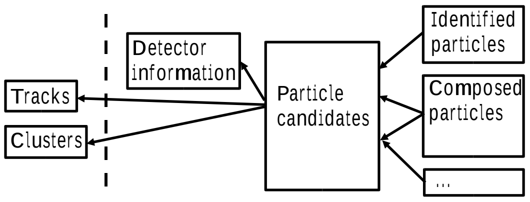

Figure 1: Structure of the μODS with pointers from identified and composed particles to the particle candidates and from the candidates to the detector information on μODS and ODS.

The production of the μODS and HAT from one DST or ODS (typically comprising 50k events) presently takes about one hour. It is thus possible to reproduce a whole year worth of μODS and HAT files within a few days. New μODS and HAT are centrally produced whenever new analysis algorithms or calibration results are available.

Data Access

Data access is implemented in a set of skeleton classes. The implementation of these classes followed two main requirements:

Firstly, a transparent access to the different levels of the data: 'Smart' accessor functions retrieve information about event and particle attributes (also across boundaries of different files, e.g. μODS-to-ODS). Secondly, the access to the data is partial, i.e. particular pieces of information are retrievable by direct access. The access is performed only when this piece of data is effectively needed.

These requirements are implemented in the class 'H1Tree' (and a number of helper classes) by using different RooT trees in parallel, one for each storage level. The user may also add additional trees of private per-event-information ('user trees').

The loop over the events is done by an iterator delivered by the H1Tree. The information about filenames, run and event ranges, and physical layout of the data is stored in a small MySQL database called the RunCatalog. The user just asks for a certain fraction of the H1 events, for example by specifying a run range, a run/event range, a period of data taking. The H1Tree (via the tree iterator) delivers the requested events directly to the user. In addition, a specific event selection can be specified using the information stored in the HAT. For further iterations over the same events the locations of the delivered events are stored in a RooT event list (TEventList).

Within a given event, the direct access, e.g. from a particle candidate object (μODS) to its original track object (ODS) is organized by a persistent container class relation, H1Pointer, which also takes care of the reading of the requested pieces of information from a file.

Physics Algorithms

To ensure quality and extendibility of the new analysis software, a modular organization of loosely coupled physics algorithms is essential. The aim is to allow for routines developed in particular user analyses to be integrated in the official production code, furthermore referred to as 'filling code'. Modularity and portability is a prerequisite for the goal that the best knowledge of all physics working groups in H1 be propagated into one common framework, i.e. the new analysis framework.

The interface between the filling code and the physics algorithms is structured such that the addition of new algorithms involves minimal changes to the software. Technically the algorithms are implemented in separate classes that obey the same interface (polymorphism). For each of the identified particles a separate algorithm is run.

For the H1 Experiment, as a running experiment with a continual flow of incoming new data and outgoing physics results, it is essential that the quality and precision of the physics analysis be sustained. Therefore a migration scheme has been adopted in which the old physics analysis software, which is well understood in terms of quality and performance, is used in areas where the new object-oriented code is not available yet.

Backward compatibility has been aimed for from the very beginning of the project. This makes it possible to further perform analyses in the old scheme, albeit loosing some of the improvements due to a necessary on-the-flight conversion to the old data structure.

A number of analyses has already started, e.g. using the newly implemented jet finder and classes for identification of J/ψ and D^* mesons. These analyses are well underway and results can be expected soon.

Event Display

A new event display has been developed. It is fully integrated in the new analysis framework and thus allows for the direct dialogue between the analysis part (e.g. event selection and histogramming) and visual inspection of events.

The display was originally derived from the Alice 3D RooT display, inheriting from the modern GUI and underlying graphics. Thanks to the RooT Run Time Type Information (RTTI), objects on the screen can be picked and inspected, and one can follow links to the physics information. Also graphics sliders can be used to interactively change the image such that only objects relevant for a certain analysis are displayed.

The new event display is fully backward compatible to the previous 2D command-line based program based on 'LOOK' [2]. The implementation reuses existing code containing expert knowledge about detector details, thus fully integrating the functionality of the previous display. A new parser for the 'LOOK'-macro language was written in C++. The new program combines modern features, such as the GUI, the click and inspect options and the 3D-display which is fully embedded in the object-oriented analysis environment with the advantageous features of the old display.

Summary

The H1 Collaboration has successfully introduced a new analysis framework, based on object-oriented programming techniques.

The key to the success was the clear definition of the scope of the project: The introduction of new object-oriented software is restricted to the physics analysis tools and user code. It goes along with the broad introduction of the RooT-package in many other HEP experiments. The existing reconstruction and simulation code remains untouched. At present heavy use is made of the well-established precision physics algorithms, as implemented in Fortran. These algorithms are being replaced by object-oriented algorithms as they become available. The physics working groups develop algorithms and add their code via well defined interfaces. Backward compatibility of the data stored in the new framework with the old DST allows to slowly phase out existing analyses, such that no interruption of the flow of physics results occurs. The end users obtain a nice and easy-to-use product integrating all analysis specific tools into one single framework.

The H1 Collaboration is convinced that physics analysis will greatly profit from the new superior analysis environment and enhanced physics performance.

References

[1] V.Blobel, The BOS System - Dynamic memory management, DESY Internal Report R1-88-01 (1988)

[2] V.Blobel et al., Look Program Manual, H1 Internal Note

[3] http://wwwinfo.cern.ch/asdoc/hbook_html3/hboomain.html

[4] http://wwwinfo.cern.ch/asd/paw/index.html

[5] http://root.cern.ch/

[6] http://root.cern.ch/root/Cint.html

[7] H1 Collaboration, The H1 detector at HERA, Nuclear Instruments and Methods in Physics Research A 386 (1997) 310-347

[8] T.Benisch et al, 'A New Data Storage Model for H1', Proceedings of CHEP 2000