The ATLAS JetTagging Event Data Model

Andreas Wildauer, CERN PH-ATC and University of Innsbruck, on behalf of the ATLAS Inner Detector Software Group
Andreas.Wildauer@cern.ch

Introduction and Motivation

For many physics analyses in ATLAS, reliable and efficient jet tagging algorithms are important. A general jet tagging framework and its Event Data Model have been developed and implemented into the ATLAS Athena offline reconstruction software. One of the guiding design principles of these jet tagging packages is a strong focus on modularity and defined interfaces using the advantages of the new ATLAS Event Data Model and object oriented C++.

The benefit for the developer is modularity in terms of easy expandability of the tagging software with additional and different tagging algorithms. The user profits from a common look and feel of all algorithms and also from an easy configurable jet tagging chain: turning on/off different taggers, retagging on the fly just before the analysis and combining the results from various taggers with hindsight during the analysis.

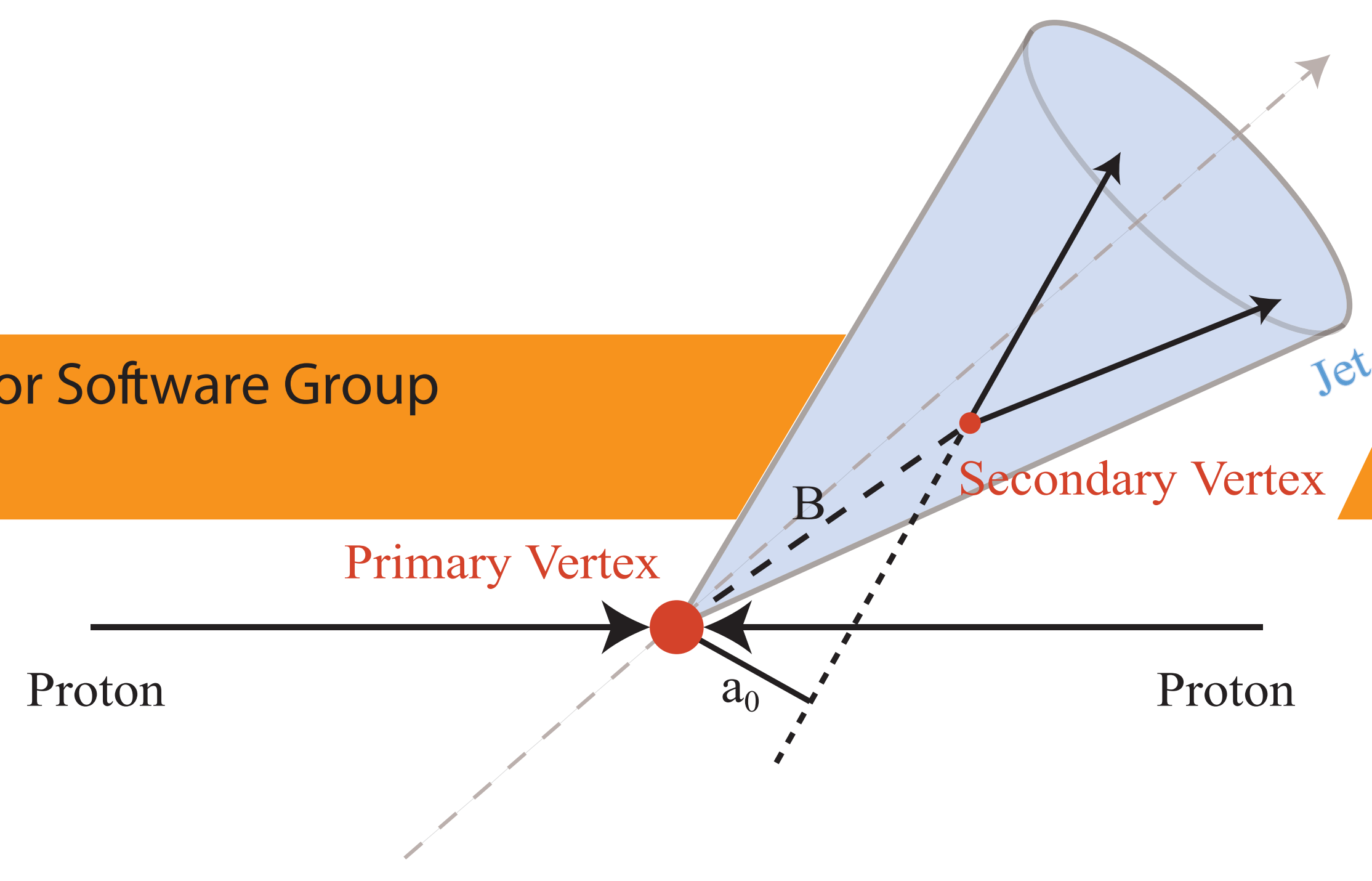


Fig 1
Schematic representation of a proton-proton collision. The outgoing quarks hadronise into a bunch of collimated particles called jets. The type of particle (quark, tau, ...) which caused the jet is of interest for many physics analyses. In this case the jet stems from a heavy b quark resulting in a jet topology containing a secondary vertex and tracks with large impact parameters.

Jet Tagging Process

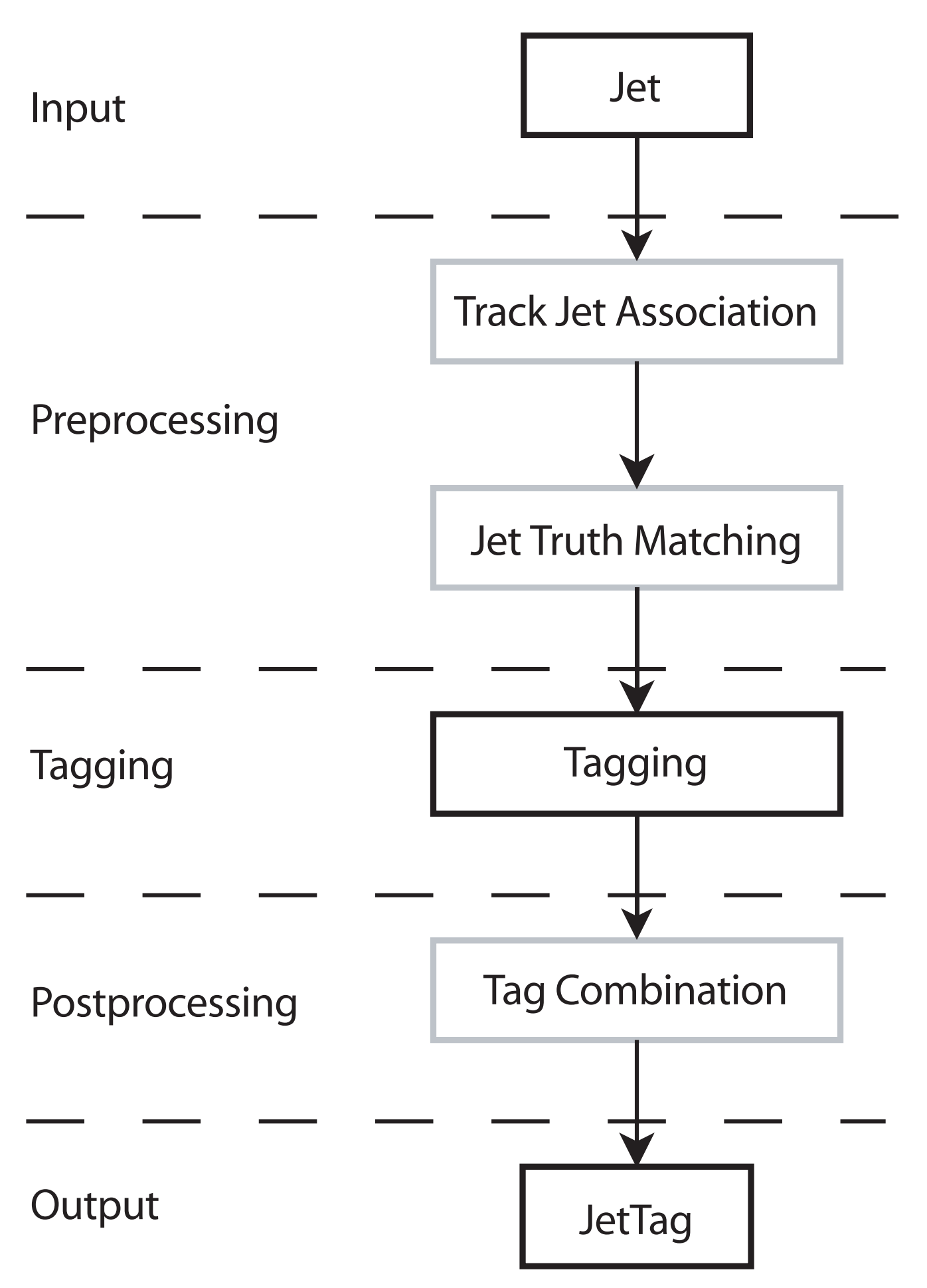


Fig 2
The jet tagging process is split into several steps: Preprocessing of input jets, tagging of jets and combining the results (postprocessing).

The task of a general jet tagging framework is divided into several steps. First of all, the framework needs to define a common Event Data Model for the tagging process. This object has to be capable of storing the results of existing taggers as well as those of future tagging algorithms. Secondly, the framework needs to define common interfaces for all jet tagging algorithms such that they can be executed by a higher level algorithm in any possible order and number. In conjunction with the Event Data Model, this common interface guarantees a modular and extendable jet tagging environment. Thirdly, the framework needs to provide a set of helper tools which can be used by the top level or the tagging algorithms to fulfill common jet tagging tasks. Among these tasks are truth matching of input jets, association of tracks to input jets, input/output of reference histograms and calculation of a discriminating variable.

Event Data Model

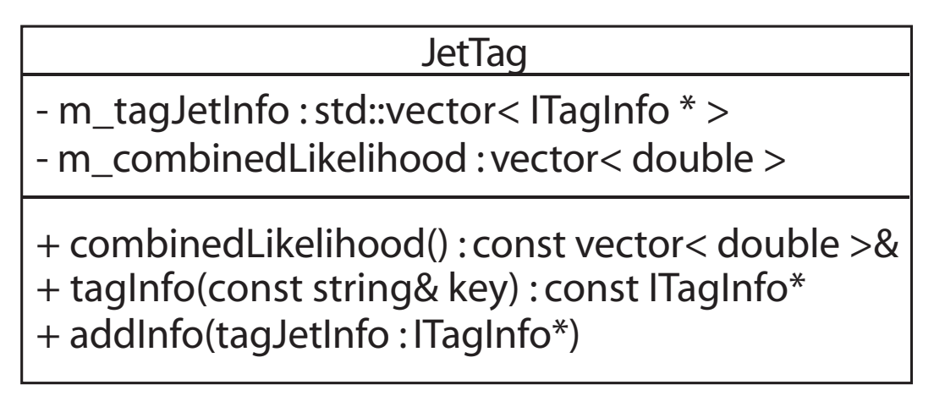

JetTag
- m_tagJetInfo : std::vector< ITagInfo * >
- m_combinedLikelihood : vector< double >
+ combinedLikelihood() : const vector< double >&
+ tagInfo(const string& key) : const ITagInfo*
+ addInfo(tagJetInfo : ITagInfo*)

Fig 3
The JetTag object stores all relevant tagging information. Results produced by individual tag algorithms is stored in a vector of ITagInfo objects.

The ATLAS Event Data Model defines data objects which are used during event processing. Tracks, jets and calorimeter cells are some examples.

The JetTag object is the dataobject which stores all information concerning the tagging process. Modularity of the jet tagging software requires this object to be flexible and extendable by design.

This is achieved with "info objects". The ITagInfo base class defines the interface and offers space for common tagging information: tag weight and likelihood.

Information unique to a specific tagging algorithm is stored in derived classes.

The common interface of these objects allow other tools to combine the tag information of several taggers to a more powerful one.

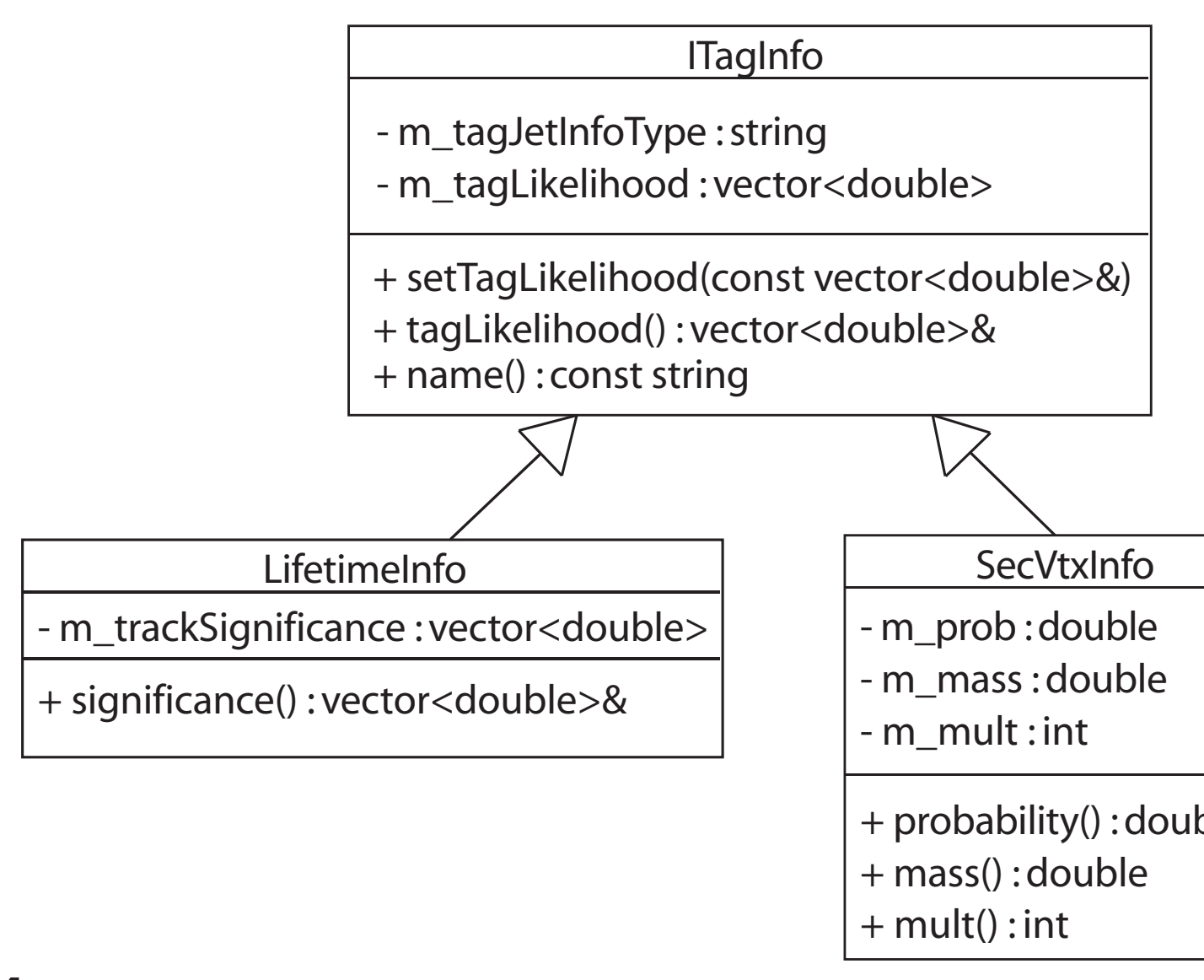

ITagInfo
- m_tagJetInfoType : string
- m_tagLikelihood : vector<double>
+ setTagLikelihood(const vector<double>&)
+ tagLikelihood() : vector<double>&
+ name() : const string
LifetimeInfo
- m_trackSignificance : vector<double>
+ significance() : vector<double>&
SecVtxInfo
- m_prob : double
- m_mass : double
- m_mult : int
+ probability() : double
+ mass() : double
+ mult() : int

Fig 4
Every tagging algorithm fills an extended version of an ITagInfo object with common and special tag information. At the end it appends the object to the ITagInfo vector of the JetTag object.

JetTagging Interface and Sequence Diagram

A modular jet tagging sequence is achieved by leting all tagging algorithms inherit from a common base class. This class defines the interface which is used by a top level algorithm to execute all tagging algorithms.

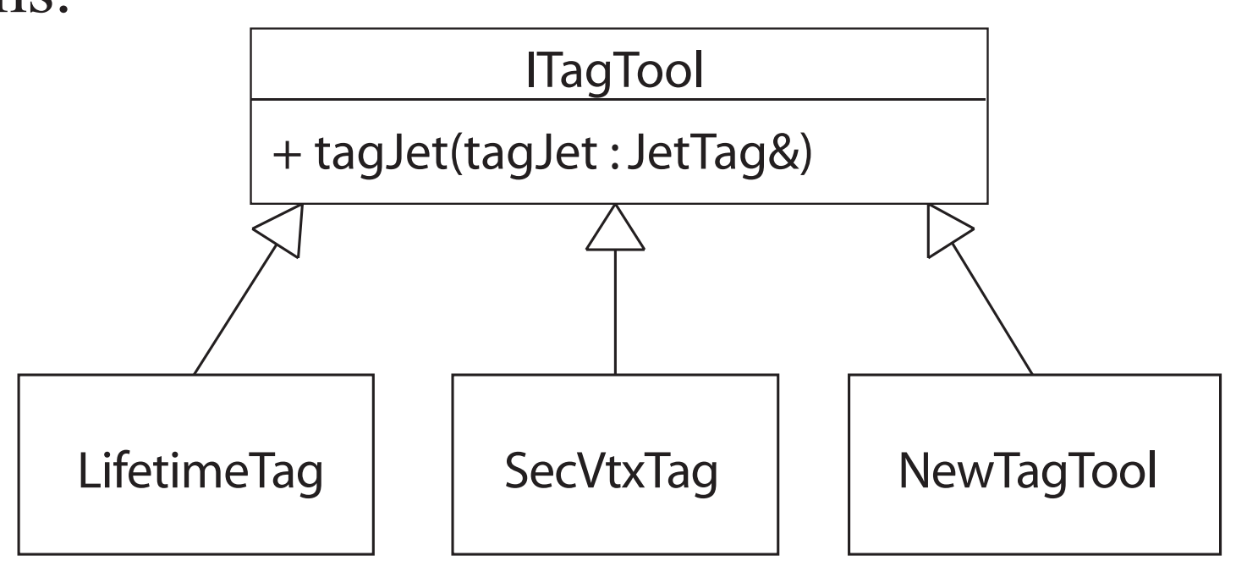
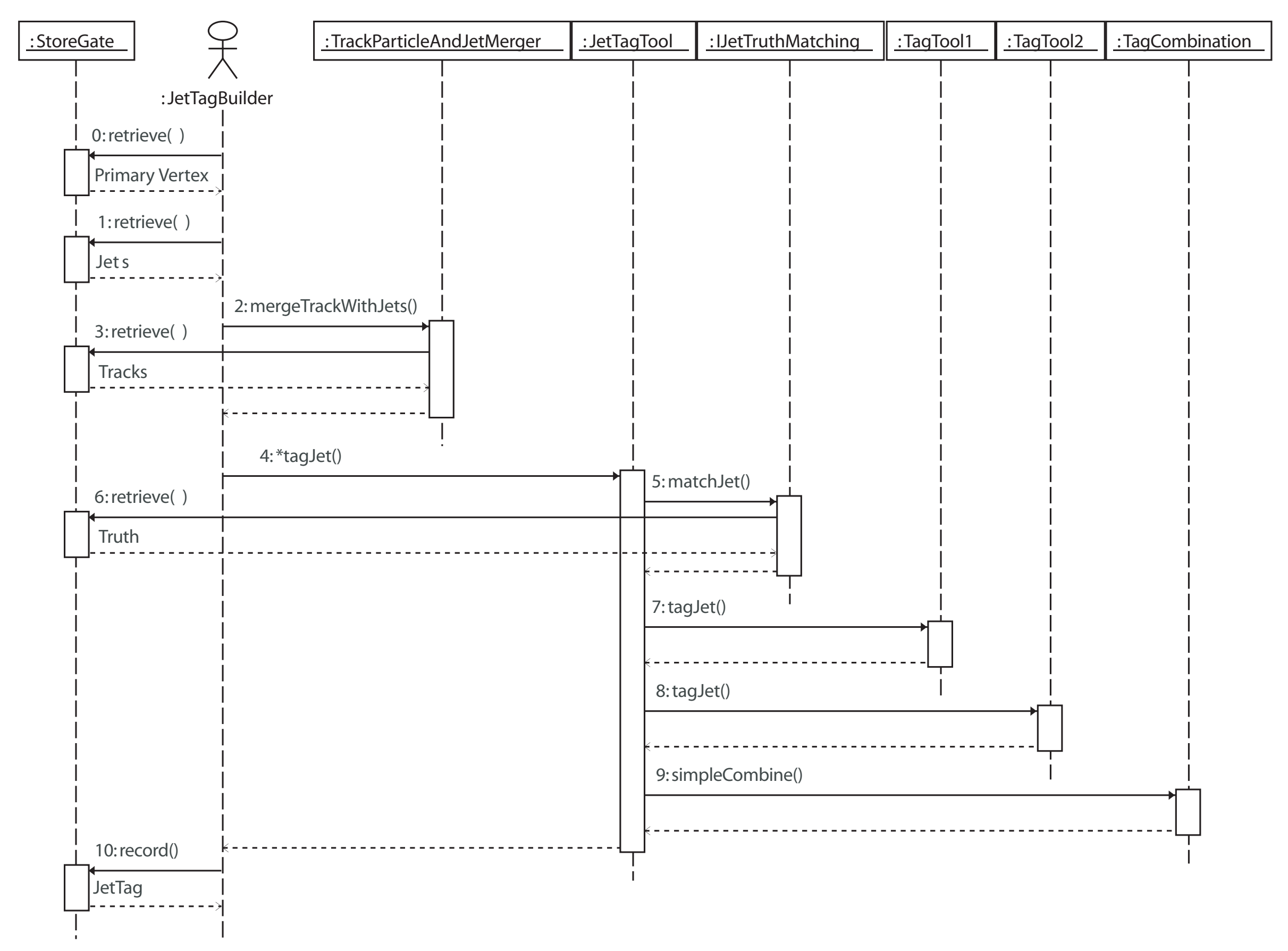


Fig 5
The common interface of all jet tagging tools is given to them by a common base class. It provides only one method which is used by the top level jet tagging algorithm to call all tag tools.

Fig 6
Sequence diagram of the jet tagging process. The top level JetTagBuilder steers the whole tagging process. It calls the tools for truth matching, track-jet association and combines the single results of the tagging tools to a more powerful discriminant.

It can execute and arbitrary number of tagging algorithms in an arbitrary order due to the use of a common Event Data Model and a common interface.

At the end of the tagging process the JetTagBuilder stores the results to be used by other algorithms and for permanent storage on disk.



JetTagging Example: b-Tagging

All ATLAS b-tagging algorithms are implemented within this JetTagging framework. Several b-tagging algorithms are run sequentialy by the top level BJetBuilder. Among them are 1D and 2D impact parameter taggers, secondary vertex based taggers and soft-lepton taggers. They all produce common tagging information (e.g. weight) but are also free to store any specific information which is relevant for this tagger.

In case of b-tagging the signed impact parameter is often used to construct a discriminating variable (Fig 7). The JetTagging framework also offers tools to write and read the histograms of these variables and to construct a weight or likelihood (Fig 8) which can later be used by the physicist to distinguish between b jets and background jets. The quality of the b-tagging is measured in terms of rejection of background jets for a given selection efficiency of signal (i.e. b) jets. Common benchmark figures are a light jet rejection of 100 with a selection eficiency of 60% (Fig 9).

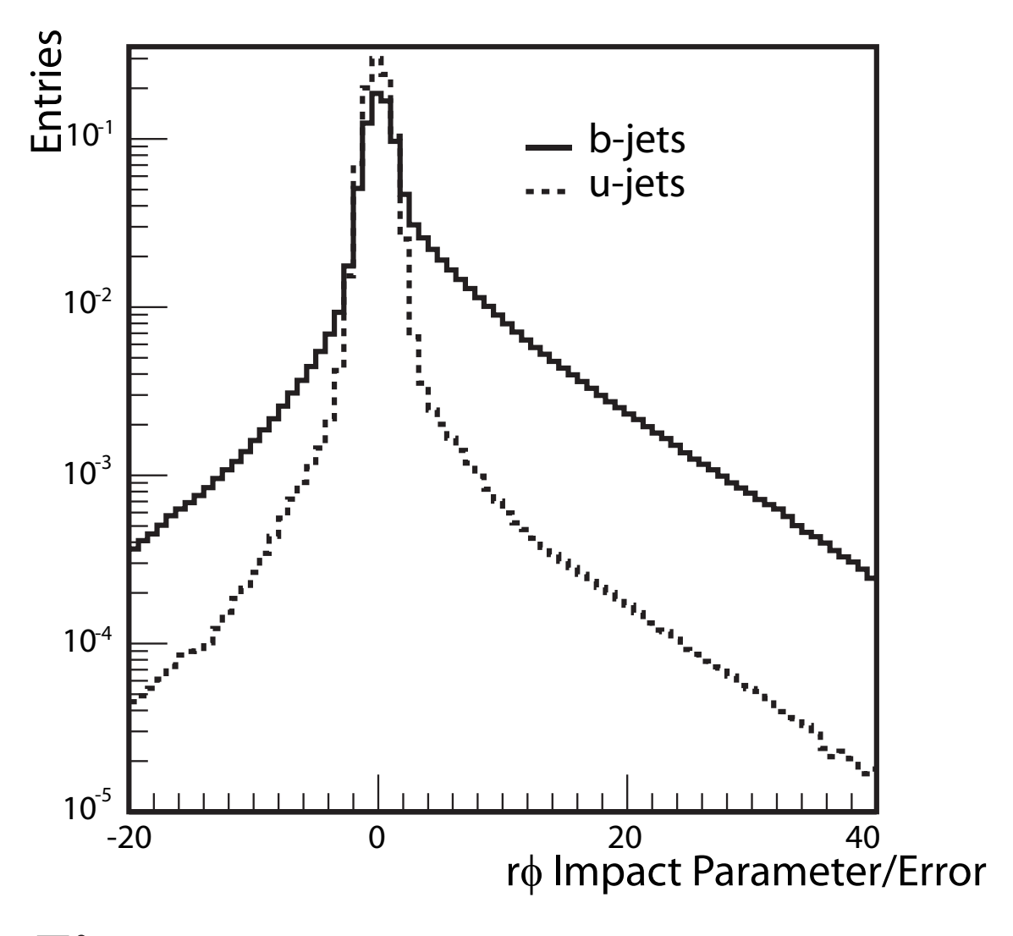
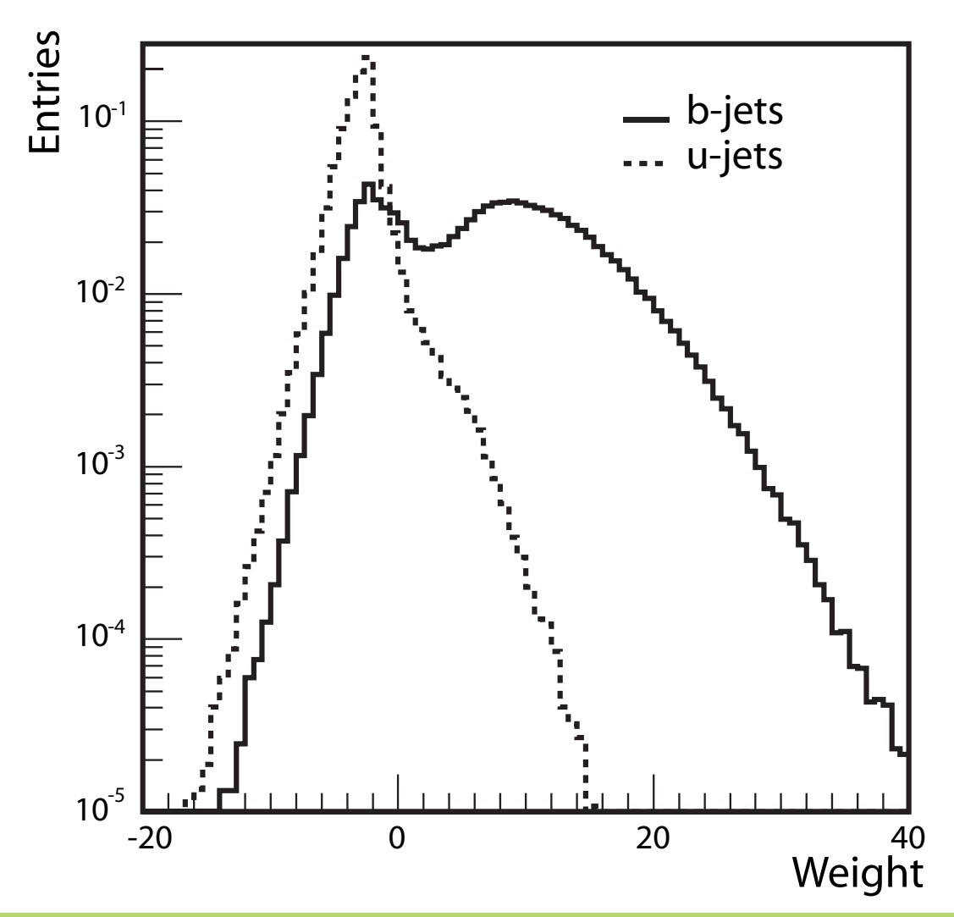


Fig 7
Impact parameter divided by measurement error for b jets and light jets.

Fig 8
Weight distribution for b jets and light jets.



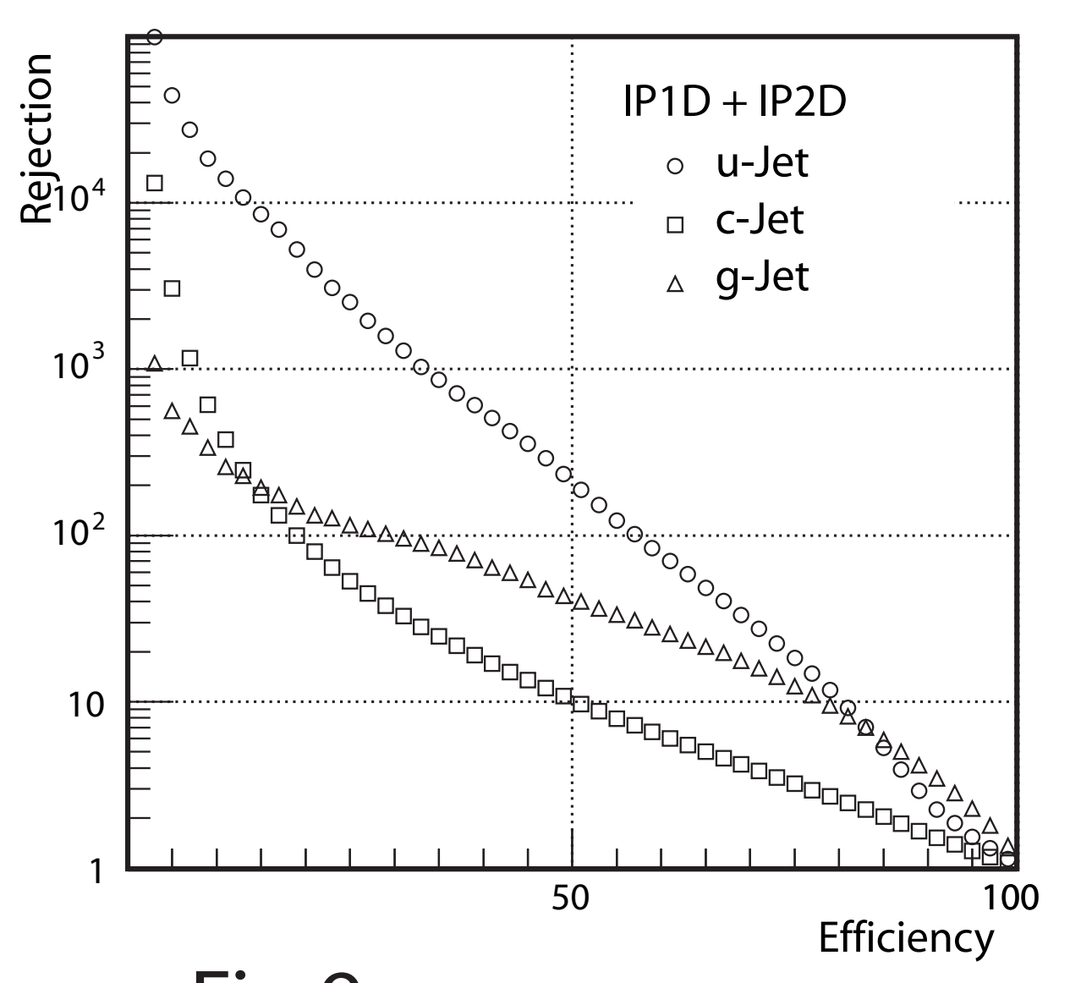


Fig 9
Rejection of light jets, c jets and gluon jets as a function of tag efficiency.