A software framework for KM3NeT

Claudio Kopper, Thomas Eberl, Alexander Kappes
for the KM3NeT Consortium

Erlangen Centre for Astroparticle Physics (ECAP)
Universität Erlangen-Nürnberg
Erwin-Rommel-Str. 1
91058 Erlangen, Germany

***Abstract*. Modular software frameworks have become indispensable for large-scale experiments like the KM3NeT km^3-scale water Cherenkov neutrino telescope, which is currently in the planing phase. After an introduction to the concept of a software framework, the adaptation of *IceTray*, a framework developed by the IceCube collaboration, for the KM3NeT detector is presented. Framework features specific to this detector are highlighted and an overview of the currently available modules is given.**

***Keywords*: neutrino astronomy, software framework, KM3NeT**

I. Introduction

A major physics experiment like the KM3NeT neutrino telescope [1] requires an immense amount of hardware development effort. However, to make use of the telescope, also software that allows running the experiment and, most importantly, to transform the raw data to high-level physics information is needed. Such software cannot be written by a single person and thus has to be created in a collaborative effort.

A software developer should not need to know every single line of code of a particular program to be able to extend it. Most of the time, the task is to implement or change a single algorithm only, without modifying the rest of the reconstruction software. It is thus necessary to write modular code with a predefined data flow from the very beginning. Such program modularisation and the definition of a corresponding data flow are the essential elements provided by a *software framework*.

In this article we give a definition of a software framework and illustrate why frameworks have become indispensable for major physics experiments. Afterwards, the *IceTray* framework [2], developed and used by the IceCube collaboration [3], is introduced. Due to its features, which are already targeted at neutrino telescopes, IceTray has been chosen as the software framework for KM3NeT as well. Its adaption to KM3NeT is discussed and an overview of the currently existing software modules is given.

II. Definition of a framework

A possible definition of a software framework can be: a software framework is a set of rules, interfaces and services provided to a software developer, who can use it to perform a set of tasks.

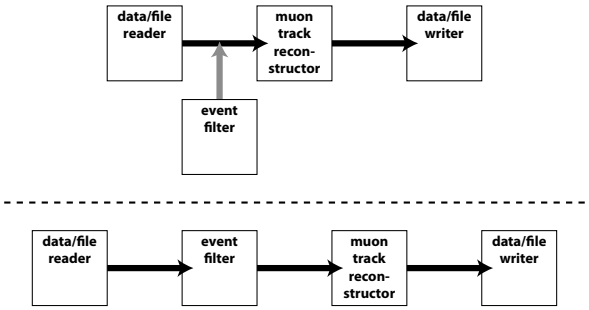

Fig. 1. Simplified reconstruction chains consisting of *modules*, with data flowing from one module to the next one. The first chain consists of three modules. In a framework, tasks like adding modules can easily be performed. In this example, an event filter is inserted at a particular point in the first chain. The resulting reconstruction chain is shown below the dashed line.

In a modular framework, the user should only have to change a few lines in a steering file to modify an analysis chain. This is illustrated in Fig. 1: An existing analysis is changed by adding a further module.

Frameworks should not be confused with class libraries. A class library, like a framework, provides a set of interfaces and services. It does not, however, enforce a particular program structure and leaves it to the individual programmer's code to specify the structure. A framework defines an overall program pattern, which makes it easy to extend: a developer implements algorithms, whereas the framework specifies how their implementations interact.

Typically, frameworks enforce this overall design. At the same time, they avoid being too restrictive, to allow for the necessary freedom to developers and end-users.[1] This is achieved by using a modular approach, where each subsystem can be modified, added or replaced without altering the others.

Similar to a computer operating system, a framework

[1] In an object-oriented language like C++, a particular design can be enforced by using the concept of *encapsulation*: the inner workings of a class should always be hidden from other classes. Another important concept, called *polymorphism*, allows the framework to be extended in a well-defined way: only a small set of abstract classes is exposed from the framework. These classes are then overwritten when implementing an algorithm.

only provides a high-level interface to make life easier for programmers and end-users. To be able to do analysis, the physics contents have to be added to the framework in the form of algorithms. To complete the analogy, a computer is useless if it is just installed with a bare system kernel. It can only be used after installing applications like a web browser or an office suite. These applications then make use of the hardware by accessing the abstraction layer provided by the system kernel. Of course, in principle, it is possible to use a computer without an operating system by creating an application that accesses all hardware directly. But clearly, such an approach would neither be modular nor easy to use or develop.

Using a framework saves the physicist a lot of time in learning how to use a particular software package every time s/he wants to perform a new task on a set of data. All algorithms are implemented in modules having the exact same structure. This makes the code easier to read and understand. The software user can thus start to work on physics problems with a significantly shorter learning period.

A modular structure also leads to more flexibility. Systematic analyses can be performed more easily, as the high-level program flow and all fine-grained module parameters are always visible to and changeable by the framework user.

Frameworks facilitate the collaboration between different groups doing software development. As all modules have the same format, they can be easily exchanged. This is especially advantageous when doing cross-checks and also helps implementing an overall quality control scheme: Code that is broken down into small independent units can be more easily reviewed by others than a single monolithic program.

III. IceTray

IceTray, the software framework developed and used by the IceCube collaboration [3], provides all the features mentioned in the previous section. Developers create *modules*, which can be dynamically linked into the framework. They contain the actual algorithms responsible for event reconstruction or simulation. Data is passed from module to module in data containers called *frames*. Each frame typically describes a single event. A frame consists of a list of name-object tuples. Except for an I/O method, there are no requirements imposed on the objects stored in a frame. This design decision makes the framework easily extendable.

The framework already provides modules for reading in and writing out data. These modules can be used at any time in the reconstruction chain, so that data can be written out at any stage. Though IceTray uses its own data format based on routines from the *boost.org* libraries [4], it is possible to write reader modules for any other data format. This has been demonstrated by providing a reader module for standard ANTARES ROOT-files as produced by the current ANTARES online data acquisition (DAQ) system [5]. It should be noted that, in principle, the framework does not require to write out any intermediate data, as data input and output is done in dedicated modules. A full Monte Carlo simulation of events and their reconstruction can thus be done in IceTray without ever writing temporary files to disk. Of course it is still possible to produce intermediate data files. Data can be written out at any point in the analysis chain at the user's discretion.

The order of modules can be defined when initialising the framework. This is typically done using the *Python* scripting language [6]. In addition to Python, compiled C++ code can be used to control IceTray module chains.

IceTray comes with *dataclasses*, containing class definitions usable for storing positions, directions, particle tracks, optical module (OM) properties and all other data necessary during event reconstruction and simulation. These classes are usable for KM3NeT with only minor modifications. These modifications are mostly necessary because the developers of IceTray assumed that all OMs would look either vertically downwards or upwards. This is true for IceCube, but assumedly not for KM3NeT.

A few simple non-physics modules, such as a generic event selector template and a ROOT tree writer can also be re-used for KM3NeT. None of these helper modules contain any physics-relevant code.

IceTray includes an installation system for almost all external packages it depends on, like ROOT and several other libraries. The installation is done using a variant of the *ports* installation system [7]. It substantially simplifies this process, as all packages can be downloaded and compiled with only two shell commands.

IV. Adaptation of IceTray to KM3NeT

To enable IceTray to work with the data that is currently available, a considerable fraction of the code from the ANTARES experiment [13], one of the KM3NeT pilot projects, was modularized. Most of these modules can also be used for KM3NeT without major changes. In addition to that, new modules have been implemented which are targeted specifically to the simulation of different proposed KM3NeT detector designs. Figure 2 shows all modules mentioned in this section and illustrates in which way data flows between them.

In addition to the DAQ file reader module mentioned above, a module that permits reading the standard ANTARES Monte Carlo file format was developed. It enables the user to read data produced by the standard ANTARES Monte Carlo packages [8]. Thus, all of the currently existing Monte Carlo data can be used with IceTray.

Events can not only be produced by the ANTARES Monte Carlo packages, but also by using other event generators like *ANIS* [9] (neutrino events) and *CORSIKA* [10] (atmospheric muons). Both of them are also in use by IceCube and already exist as IceTray modules. Once events have been generated, the resulting particles need to be tracked through the detector volume and

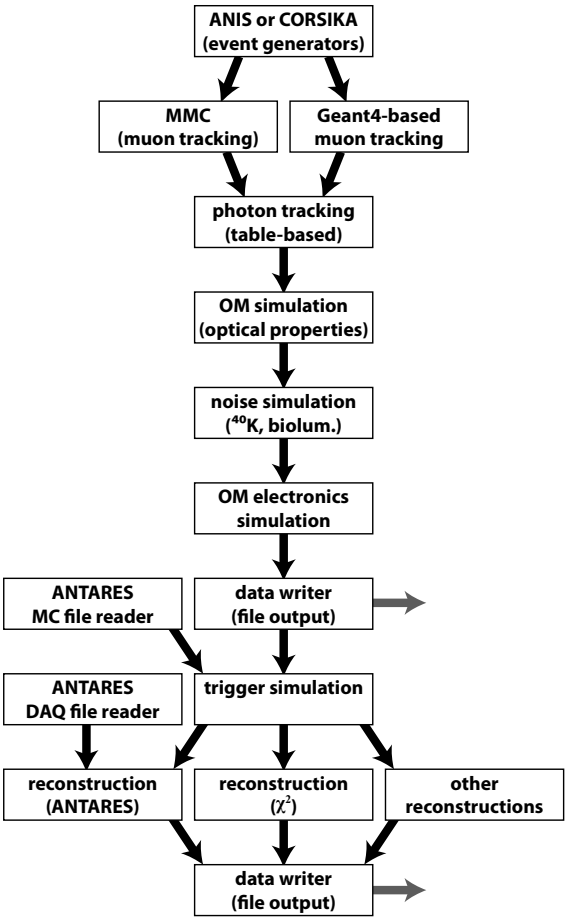

Fig. 2. Example simulation and reconstruction chain consisting of all modules discussed in section IV. The arrows show how these modules can be connected to form a full chain, where two arrows connected to a single module denote alternative module chains. Also shown are two places where old ANTARES Monte Carlo or ANTARES DAQ data could be read in. In these cases, the file reader modules would replace all previous simulation modules. In this example chain, data is written out at two points: once after the readout simulation and a second time after the final reconstruction step. This is, of course, fully optional.

their Cherenkov light needs to be simulated. Muons, being the main detection channel of neutrino telescopes, can either be tracked by the *MMC* [11] package which was modularised by IceCube or by a custom tracking code based on Geant4 [12] developed specifically for KM3NeT. The Geant4-based module has the advantage that it can also perform a full tracking simulation of the muon and all its secondary particles including their Cherenkov light. This feature would not be useful in a standard simulation, as it is rather slow. It can, however, be used for cross-checks of faster, parameterised or table-based simulations.

Such a simulation, which takes care of generating and propagating Cherenkov light to each OM has been developed. It contains a table-based fast simulation code which includes full light absorption and scattering in water. To allow for more flexibility during the detector design phase, the light propagation is performed in two steps: first, the simulation records all photons arriving at a sphere around the OM. In a second step, photons are propagated into the actual OM and to the surface of the photomultiplier tube (PMT) where they then produce actual hits. This two-step simulation allows for studying different OM designs without having to re-run the time-consuming light propagation process each time. Environmental background hits (from ^{40}K decays and bioluminescence) at the detector site [14] are generated with a fixed user-configurable rate per OM. Alternatively, the noise rate per OM can be taken from real data.

To be able to compare Monte Carlo events to real data, full simulations of the OM's digitisation electronics are needed. The code for these simulations has been encapsulated into modules: The OM simulator uses a parameterised approach to simulate the transit time spread and the amplitude smearing of the PMTs. The simulation of the PMT signal digitisation is fully compatible with the one currently in use by ANTARES, but flexible enough to be used with other readout designs.

When using Monte Carlo data, it is necessary to simulate different detector trigger conditions. For this purpose, the complete ANTARES core trigger code was encapsulated into an IceTray module. This module receives the current event data and converts it into the format used internally by the ANTARES trigger. The original trigger code can thus be used unchanged, which allows for easy maintenance and updates in case new trigger algorithms are added to the ANTARES DAQ. After calling the trigger routines, the module converts all results back to a data format compatible with IceTray and passes all information to the next module.

Reconstruction of events is typically done using a simple pre-fit algorithm in conjunction with a more sophisticated reconstruction algorithm, possibly combined with an iterative series of hit selections. Therefore, such reconstruction strategies can be broken down into a number of modules. This has been done for the standard ANTARES muon reconstruction strategy [15]: All pre-fit, intermediate fit and hit selection steps have been re-written as IceTray modules. Track candidates are passed from each fit module to the next one, where they can be used as starting points. The resulting track candidate is then passed to a final fit algorithm which was also converted into an IceTray module. Other available reconstruction modules include a hadronic cascade reconstruction [16], a fast χ^2-based code and a reconstruction targeted specifically to a *MultiPMT* design [1], where one optical module contains a larger number of small phototubes. Thus, the framework user already has an extensive choice of reconstruction algorithms, tailored specifically to different detector layouts.

The simulation and reconstruction chain laid out in this section is currently in use by KM3NeT for studies aiming to determine an optimal layout for the final detector design.

V. SUMMARY

Software frameworks are essential parts of every large scale physics experiment. KM3NeT recently adopted the *IceTray* framework as their official software framework during the KM3NeT design study phase [17]. This has the benefit that Monte Carlo studies done now will be performed in the same environment as the real reconstruction later on, which leads to results that are comparable more easily.

Another important point is the steeper learning curve for the developer in learning how to use a particular software. As the program structure in a framework always stays the same, the programmer has to learn it only once in the beginning.

Using IceTray in particular has the advantage that this framework is already in use for neutrino telescopes, is actively developed and has been thoroughly tested. Using the same data-format as IceCube also opens future possibilities for sharing data and exchanging high-level analysis code.

A considerable amount of work has been invested to make IceTray compatible with ANTARES data and to modularise the most commonly used pre-existing algorithms. Additionally, new reconstruction algorithms targeted specifically to KM3NeT have been implemented. A full simulation and reconstruction chain is available and ready for use.

VI. ACKNOWLEDGMENTS

The authors wish to thank the IceCube collaboration for their co-operation and especially for providing KM3NeT with their software framework. This work has been funded by the EU in the framework of the KM3NeT Design Study, FP6 contract 011937 and by the BMBF, project number 05CN5WE1/7. A. Kappes acknowledges support by the Marie-Curie OIF program.

REFERENCES

[1] KM3NeT Consortium, P. Bagley et al., *KM3NeT Conceptual Design for a Deep-Sea Research Infrastructure Incorporating a Very Large Volume Neutrino Telescope in the Mediterranean Sea*, 2008, available from http://www.km3net.org/CDR/CDR-KM3NeT.pdf

[2] IceCube Coll., T. DeYoung, *IceTray: A Software Framework for IceCube*, International Conference on Computing in High-Energy Physics and Nuclear Physics (CHEP2004), 2004, available from http://www.chep2004.org/

[3] IceCube Coll., J. Ahrens et al., *Preliminary Design Report*, 2001, available from http://www.icecube.wisc.edu/science/publications/pdd/

[4] available from http://www.boost.org/

[5] ANTARES Coll., J. A. Aguilar et al., *The data acquisition system for the ANTARES neutrino telescope*, Nucl. Instrum. Meth. A **570** (2007) 107 [arXiv:astro-ph/0610029]

[6] see http://www.python.org/

[7] see http://www.macports.org/

[8] D. Bailey, *Monte Carlo tools and analysis methods for understanding the ANTARES experiment and predicting its sensitivity to Dark Matter*, PhD thesis, Wolfson College, Oxford, 2002

[9] A. Gazizov and M. P. Kowalski, *ANIS: High energy neutrino generator for neutrino telescopes*, Comput. Phys. Commun. **172** (2005) 203 [arXiv:astro-ph/0406439]

[10] D. Heck, G. Schatz, T. Thouw, J. Knapp and J. N. Capdevielle, *CORSIKA: A Monte Carlo code to simulate extensive air showers*, Forschungszentrum Karlsruhe Report FZKA 6019 (1998)

[11] D. Chirkin and W. Rhode, *Muon Monte Carlo: A high-precision tool for muon propagation through matter*, arXiv:hep-ph/0407075.

[12] S. Agostinelli *et al.* [GEANT4 Collaboration], *GEANT4: A simulation toolkit*, Nucl. Instrum. Meth. A **506** (2003) 250.

[13] ANTARES Coll., *Technical Design Report of the ANTARES 0.1 km^2 project*, 2001, available from http://antares.in2p3.fr/Publications/

[14] ANTARES Coll., P. Amram et al., *Background light in potential sites for the ANTARES undersea neutrino telescope*, Astropart. Phys. **13** (2000) 127 [arXiv:astro-ph/9910170]

[15] A. Heijboer, *Track Reconstruction and Point Source Searches with ANTARES*, PhD thesis, Universiteit van Amsterdam, 2004

[16] R. Auer, *Reconstruction of hadronic cascades in large-scale neutrino telescopes*, Nucl. Instrum. Meth. A **602** (2009) 84

[17] see http://www.km3net.org/