# ROME - a universally applicable analysis framework generator

M. Schneebeli[*], S. Ritt[†], PSI, Villigen, Switzerland
R. Sawada[‡], University of Tokyo, Japan

*Abstract*

This paper presents a new approach to writing analysis frameworks. We describe a way of generating analysis frameworks from a short experiment description. The generation process is completely experiment independent and can thus be applied to any event based analysis.

We focus on a software package called ROME. This software generates analysis frameworks which are fully object oriented and based upon ROOT [1]. The frameworks feature root functionality, SQL database access, socket connections to GUI applications and connections to DAQ systems. ROME is currently used at PSI, Switzerland, Los Alamos National Laboratories, USA and Triumf, Canada.

## INTRODUCTION

Analysis frameworks [2] for high energy physics (HEP) experiments are typically custom built. This is due to the fact that the experiments can differ significantly from each other. Obviously building special frameworks for each experiment requires significant man power. In this paper we describe a way of generalizing analysis frameworks, based on the idea of not writing the framework code by hand but, by letting it be generated by a translation tool that transforms an experiment description into code. The translation tool can be written completely experiment independent and hence can be used for every event based data analysis. This approach saves a lot of man power since the translation tool has only to be written once and can be used for any experiment. Moreover, it also leads to optimized analysis frameworks since all experiments using the translation tool can contribute improving it.

We present such a translation tool which comes under the acronym ROME and stands for Root based Object oriented Midas Extension. It was developed at PSI, Switzerland and is currently used by the MEG experiment at PSI, the DANCE experiment at the Los Alamos Laboratory and is under test by other experiments. We explain the implementation of the generation process in ROME, which is a translation of an experiment description specified in XML files to C++ classes. Furthermore, features such as connection to DAQ systems, database access, socket connection, GUI extension and modularity are also discussed.

[*] matthias.schneebeli@psi.ch
[†] stefan.ritt@psi.ch
[‡] sawada@icepp.s.u-tokyo.ac.jp

## GENERATING ANALYSIS FRAMEWORKS

An analysis software [2] can be divided into two parts. The framework code and the analysis code. The framework code includes the whole structure of the analysis, e.g. the program flow, I/O, the data structure, the task structure etc. The remainder of an analysis software is the analysis code itself, which is the calculation code performed on the data.

The framework code can further be divided into two parts, a part which can be implemented completely in an experiment independent way and a part which is experiment dependent. The former independent part includes mainly the event loop and basic functionality of a framework. The rest of the code is experiment dependent. This experiment dependent code, however, can be summarized in an experiment description, from which the program code can be generated by a translation tool. The analysis code however, can in general, not be simplified and has to be implemented as code by the experimenter.

Therefore, we divide an analysis software into three parts (experiment independent framework, experiment dependent framework and analysis). The amount of code needed for these parts strongly depends on the size of the final program. The experiment independent part is fixed and will vary up to 60% for a very simple analysis and will decrease to a small fraction for a complex one. The other two parts both grow with the size of the experiment, though the important fact to notice is that they grow at more or less the same rate. Therefore, the generated part of the analysis software will always be of a similar size to that of the analysis code, regardless of the size of the final analysis software. This means that about half of each analysis software can in principle be generated.

### *Benefits of a generated framework*

Generating an analysis framework has several advantages, which are discussed in this section.

**Saves Manpower:** As we pointed out before the generation of frameworks saves a lot on man power. The translation tool has to be written once and can then be used for every event based experiment. Therefore, new experiments simply have to write an experiment summary and use the translation tool to generate the whole framework code of their analyzing software.

**Leads to better analysis frameworks:** If many experiments use the same translation tool, they all can help to improve it which will also help to improve the functionality and the quality of the generated frameworks. Especially small experiments will benefit from the experience and the manpower of larger ones.

**Leads to better analysis software:** The framework includes all code which is difficult to write from a software technical view point. Since this part is generated, the experimenter can focus on the analysis code. This is usually difficult from a physics view point but not from a software technical view point. Therefore, the experimenter can place more effort into the analysis of the experiment which finally leads to a better result.

**Ease of use:** The programming inside a generated framework is easy. All classes are generated by the translation tool, therefore the experimenter does not require object orientated programming knowledge. This ensures that non experienced programmers can contribute significantly to the analysis software.

## ROME

We implemented the idea of generating frameworks and called the software ROME. In ROME the experiment independent part is implemented as ROME classes which are mainly base classes for the generated classes. The ROME classes are part of the ROME distribution. The translation tool is called the ROMEBuilder. It translates the experiment description given in a XML file into C++ classes.

### *The ROMEBuilder*

The ROMEBuilder translates the experiment description specified in an XML file into C++ code. We have chosen XML for the description because it is probably the most used standard ASCII format and it fits perfectly for this purpose. The choice of C++ for the programming language is mainly given by the fact that the generated software depends on the ROOT libraries which are written in C++. The output of the ROMEBuilder are framework classes and task classes. Task classes are based on ROOT tasks [1] which are calculation modules. The ROMEBuilder also compiles and links the classes to an executable and writes html documentation (see figure 1). Therefore, when running the ROMEBuilder for the first time it will generate a running executable, however the executable will not include any analysis code. This has to be added by the experimenter to the predefined event methods of the task classes. The ROMEBuilder can then be run again to produce an executable which includs the analysis code. This process can be repeated until the final version of the program is reached. The ROMEBuilder will, of course, not overwrite any user code.

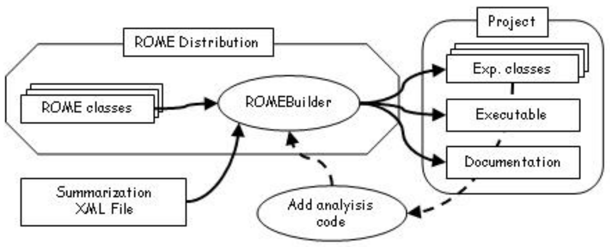

Figure 1: The translation program generates all codes of the framework from an experiment description. The framework is also linked and documented.

### *Programming inside ROME*

The generated framework is written in C++ and is completely object oriented. However, since all code except for the calculations is generated, the experimenter has only to add code to predefined functions. Therefore, he does not need to write classes himself and hence does not need to know the concept of object orientation. However, if an experienced programmer wishes to extend the framework beyond the capability of ROME he will benefit from the object orientated structure. Accordingly, ROME is designed to be very easy to use for less experienced programmers as well as easy to extend for experienced users. The framework is based on the ROOT libraries and hence the whole functionality of ROOT is available.

### *Features*

**Connection to DAQ Systems:** ROME can be used in every event-based experiment for on-line and off-line analysis. So far, only a connection to the MIDAS [3] data acquisition (DAQ) system has been implemented but it can be easily extended to other systems using a plug-in mechanism. Having a framework, which is capable of performing on-line and off-line analysis brings consistency, saves manpower and instruction time.

**Database Access:** The generated program can also be connected to a database. The access to the database is completely covered by the framework, i.e. no database calls have to be implemented by the experimenter. So far, the access to a MySQL database [4] as well as, for small amounts of data, to a XML database, has been implemented. Implementing access to any other database system is also very straight forward and can be done using a plug-in mechanism and without changing the translation program.

**Socket Connection:** ROME analyzers can share objects over socket connections to third party software. This makes it possible for example to access and display, in real time, histograms of an online analyzer in any office around the world.

**GUI Extension:** The ROME distribution also includes a GUI extension called ARGUS. The ARGUS display is generated the same way as the ROME analyzer is generated. Therefore, it can be defined in the same XML definition file as the analyzer. The final program has then three possible running modes. Firstly, it can run as a pure analyzer without any graphics extension. Secondly, it can run as an analyzer with a GUI displaying the objects of the analyzer. The third possibility is to run it as a pure display. The first mode will usually be used for an online analyzer, since it should run in a reliable way and therefore be as simple as possible. To display the online histograms in a control room or in the various experimenters offices a pure display will be used. These displays get the objects over socket connections described in the previous paragraph. For offline analysis one would typically run the analyzer together with a display.

**Modularity:** ROME generated analysis software packages are highly modular. The calculation code is split up into tasks, which contain one or several calculation steps. Tasks can be exchanged arbitrarily as long as they access the same data. In this way two different types of analysis can be performed with the same program, by rearranging the tasks. This can be done simply by configuring the program over an XML configuration file before startup and without relinking of the program.

**Distribution:** The software runs under Windows, Linux and Macintosh and can be downloaded from a subversion repository. Instructions for downloading and using the software can be found on the ROME home page http://midas.psi.ch/rome.

## ACKNOWLEDGEMENTS

We would like to thank all users of ROME for using and testing this software.

## REFERENCES

[1] R. Brun et al., http://root.cern.ch.

[2] For this paper it is important to distinguish between the term analysis framework and the term analysis software. With analysis software we mean a final software for analysing experimental data. Analysis framework means a framework for writing an analysis software, that means it refers to all code exept the analysis code.

[3] S. Ritt, http://midas.psi.ch/.

[4] MySQL AB, http://www.mysql.com.