CMD-3 PROJECT OFFLINE SOFTWARE DEVELOPMENT

A. Zaytsev*, Budker Institute of Nuclear Physics†, Novosibirsk, Russia
E. Algaer, S. Pirogov, N. Stuly, Novosibirsk State University‡, Novosibirsk, Russia

Abstract

This contribution contains the general design overview and a status of implementation of the **CMD-3** project offline software for detector simulation and event reconstruction. Software design standards of the project are object oriented programming techniques, C++ as a main language, modular approach and XML/Schema usage for configuring software components. The dedicated software development framework (Cmd3Fwk) was implemented in order to be the basic integration solution for building offline reconstruction code, simulation tools and the 3rd level trigger for the **CMD-3** detector. The framework and a set of modules are currently supported on Linux (Fedora Core 1 and 2) with gcc 3.3 compilers for x86 and x86_64 architectures. We also look forward to achieve high level of integration with the ROOT framework and Geant4 toolkit.

VEPP-2000 COLLIDER AND CMD-3 DETECTOR

The **CMD-3** is the general purpose cryogenic magnetic detector [1, 2] for the **VEPP-2000** electron-positron collider [3, 4], which is being commissioned at the Budker Institute of Nuclear Physics (BINP, Novosibirsk, Russia). The main aspects of the physical program of the experiment are the study of known and the search for a new vector mesons, study of the $p\bar{p}$ and $n\bar{n}$ production cross sections in the vicinity of the threshold and search for exotic hadrons in the region of center-of-mass energy below 2 GeV. The **VEPP-2000** collider also is going to perform the first test of round beam technique [5, 6].

The essential upgrade of the **CMD-2** detector (designed for the **VEPP-2M** collider at BINP) production farm and distributed data storage management software is required to satisfy new detector needs and scheduled to perform in near future.

CMD-3 SOFTWARE DEVELOPMENT FRAMEWORK

General Overview

The **CMD-3** Software Development and Data Processing Framework (officially named as CMD-3 SD/DP Fwk or Cmd3Fwk) is based on the following assumptions about typical HEP data treatment procedure:

- The data analysis procedure is well represented by a directed acyclic graph with the modules and data instances at the nodes, therefore the reverse call method of building self-organizing modules chain can be used.
- The input data is divided into so called ”runs” consisting of so called ”events” with the similar structure withing the certain run, thus the cycle over the events and runs for the data set being analyzed can be organized by the framework tools themselves, not by the user-written code.
- The data instances are produced by the modules and only the creator module of the certain data instance is intended to modify it, so all the intermediate stages of the data processing are preserved during the certain event reconstruction.

Core Components

The **CMD-3** framework distribution consists of Cmd3Fwk_Share, Cmd3Fwk_Core and Cmd3Fwk_MDTC shared libraries, cmd3fwk (”on demand” data processing manager) and cmd3dmtc (code templates generator).

The data set processing sequence can be started by passing modules chain XML configuration and input data sources list to cmd3fwk executable. The layout of cmd3fwk usage is shown in Fig. 1.

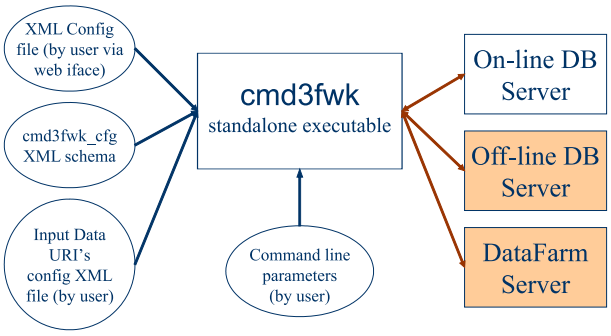

Figure 1: Cmd3Fwk core executable (cmd3fwk) configuration and usage via command line interface.

Module instances factory is created during the dynamic loading of libraries specified in the configuration and used for building of requested instances. Modules and data instances are registered in so called ”proxies” and accessed via unique string identifiers. All data instances are registered by the modules during initialization. An example of event processing modules chain is shown in Fig. 4.

* E-mail: zaytsev@star.inp.nsk.su
† Main Web Site: http://www.inp.nsk.su
‡ Main Web Site: http://www.nsu.ru

The configuration of the module includes a complete list of proxy input/output data instances parameters, external input/output links names, additional module parameters names and values, module status flags as shown in Fig. 2.

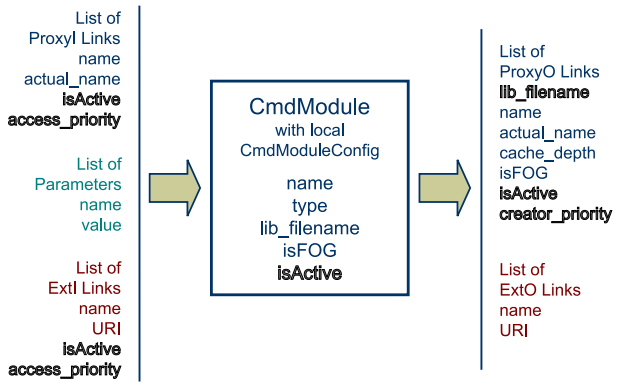

Figure 2: Cmd3Fwk module basic interface.

The code template generation tool cmd3dmtc was introduced to the framework in order to simplify the process of adding new modules by users. The user module code template and a complete code of optional intermediate classes designed to hide low level proxy interfaces could be obtained by calling cmd3dmtc executable with the modules XML description filename as one of the command line parameters. The layout of cmd3dmtc usage is shown in Fig. 3.

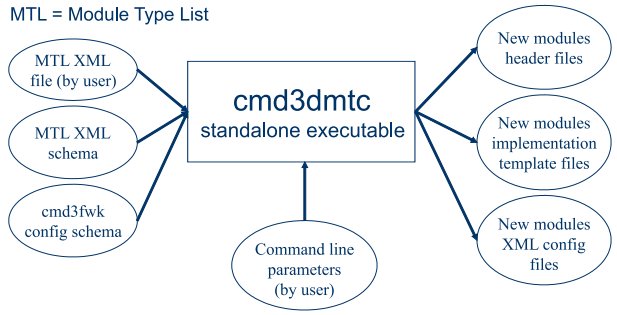

Figure 3: Cmd3Fwk code template generator (cmd3dmtc) configuration and usage via command line interface.

The low level (command line) user interfaces of core tools are suitable enough for advanced developers, nevertheless the high level web interface would be more preferred for analysis job submission and modifying of existing XML configurations by users. The technical details of implementation of such an interface are still under consideration.

Current Status of Implementation

The implementation of the core tools, configuration parsers and basic code generation library of the framework has been finished a few months ago (during summer 2004). The implementation of the common logger to be used by modules is in progress. We expect a few small changes of module and proxy interfaces in the near future due to the feedback from the users.

Framework integration with ROOT and Geant4

The simulation and reconstruction event data layout which is shown in Fig. 5 includes various external components. In order to test the compatibility of the framework with extensively used ROOT data processing framework, CLHEP library and Geant4 detector simulation toolkit we have introduced a few interface modules to Geant4 and ROOT TFile and TTree based output module as shown in Fig. 6. We observed no performance or stability issues during the test runs within that configuration.

CONCLUSION

The dedicated software development framework was implemented in order to be the basic software integration solution for building offline reconstruction code, simulation tools and the 3rd level trigger for the **CMD-3** detector.

Although the main area of application of the framework is supposed to be HEP data processing, it also can be used in any activities related to sequential data processing.

ACKNOWLEDGEMENTS

The authors are grateful to all the members of **CMD-3** online and offline software development groups for fruitful discussions during the design stage and also for useful testing and debugging feedback.

REFERENCES

[1] D.N. Grigoriev, **CMD-2** Detector Upgrade. hep-ex/0106009

[2] **CMD-3** Collaboration Web Site: http://cmd.inp.nsk.su

[3] I.A. Koop, **VEPP-2000** Project. physics/0106013

[4] **VEPP-2000** Collaboration Web Site: http://vepp2k.inp.nsk.su

[5] L.M. Barkov *et al.*, Proc. of the IEEE Particle Accelerator Conference, San Francisco (1991), p.183

[6] V.V. Danilov *et al.*, Proc of the Asian Particle Accelerator Conference, Tsukuba (1998), p.257

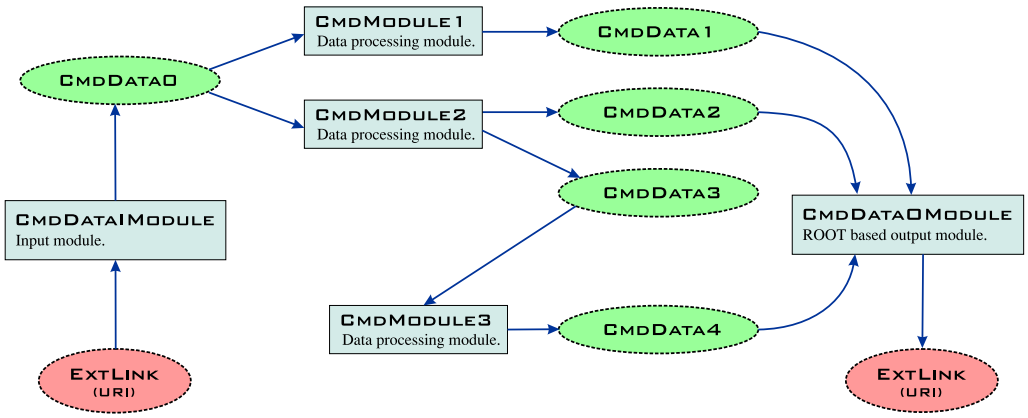

Figure 4: Example of modules and data instances dependency graph. Data instances are created by modules during the initialization. Modules called in the reversed direction from output module to input module and data instances updated only once per event reconstruction cycle. External links (ExtLink) are specified by string resource identifiers (URI) and interpreted by modules.

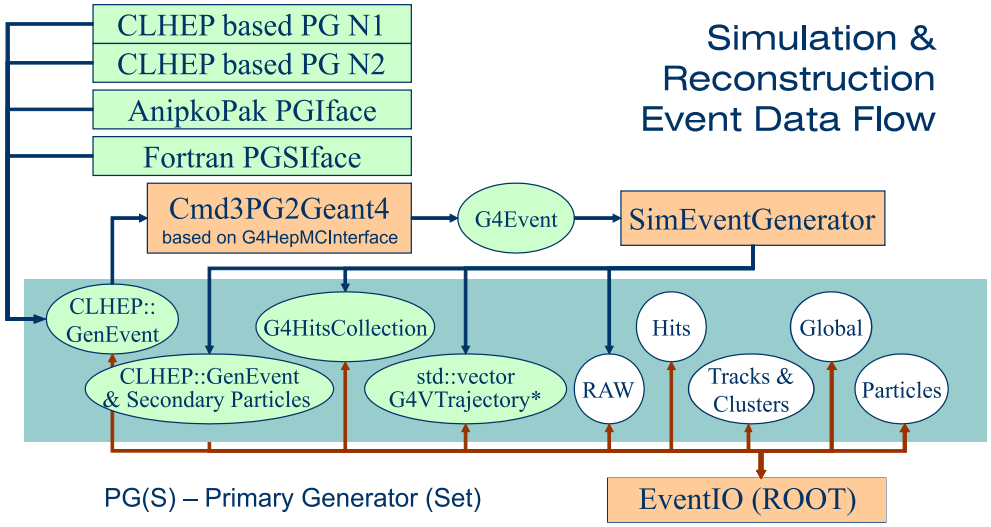

Figure 5: General layout of the data types to be used in **CMD-3** event reconstruction procedures.

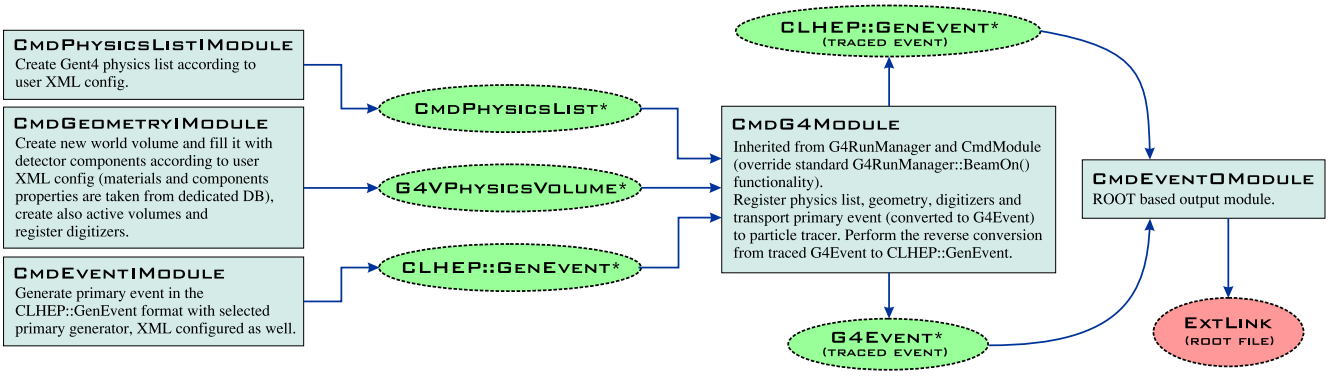

Figure 6: Cmd3Fwk and ROOT, CLHEP & Geant4 integration test configuration overview.