Design and Performance of an Object Oriented Model for CMS Track Reconstruction

Irwin GAINES [a], Thorsten HUEHN [b], Sijin QIAN [c,1]

[a] *Fermilab, Batavia, U.S.A.*

[b] *Florida State University, Tallahassee, U.S.A.*

[c] *CERN, Geneva, Switzerland*

An Object Oriented (OO) model for the CMS [1] central tracking reconstruction has been designed and coded in the C++ programming language. It has been tested with single and multiple track events and has been compared with non-OO programs. The class design of the model is based on well-known data concepts for track reconstruction in HEP, so it should be easily understood and adopted by non-expert class users. Extensive use has been made of the C++ Standard Template Library (STL) in the class design and program coding. The model incorporates many well-optimized Fortran subroutines of the Kalman filter track reconstruction package [2] in the current CMS simulation and reconstruction facility (CMSIM) as member functions of various classes. Hopefully, this model can demonstrate a possible means for a smooth transition to future object oriented programs in HEP.

Key words: algorithms, C++, mixed language programming, object-oriented methods, programming languages.

1 Introduction

As the computer industry moves to the Object Oriented (OO) era to accommodate the increasing complexity of programming tasks, we explore whether an OO model for tracking and event reconstruction can be designed by using proven data concepts. In addition to incorporating the main features of OO Programming (e.g. inheritance and polymorphism, etc.), we re-used Fortran

[1] The work has also been supported by I. Physikalisches Institut at Aachen, Germany and IPNL/CNRS-IN2P3 at Lyon, France during its course.

subroutines from the Kalman filtering track reconstruction package [2] of the current CMSIM (i.e. the CMS simulation and reconstruction facility). As those routines are highly modularized and without any common block communication with the outside world, they can easily be included as member functions of various classes.

We use the Standard Template Library (STL) [3], not only in the C++ implementation, but also in the OO model design. Some list-type classes representing vectors and linked lists (see Fig.1 of [4]) in the early versions of the OO model have now been absorbed into relevant classes by using STL containers, resulting in a much cleaner and simplified class diagram (Fig. 1a).

The whole model has been coded in the C++ program language (save for the re-used Fortran routines). C++ was chosen because of its wide use and the wide availability of commercial software products. The code was tested with single and multiple track events in the CMS central tracker and was found to be consistent with the conventional CMSIM tracking software.

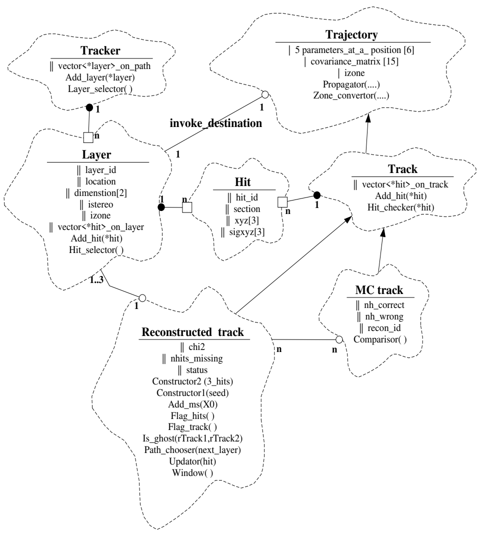
2 The object oriented model, C++ implementation and testing

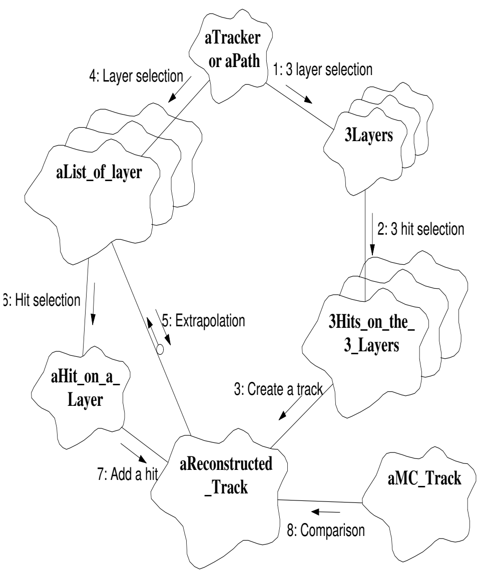
The model design process begins by formulating a "problem statement" which guides the design of the class diagram (Fig. 1a), drawn here by the commercial product Rational/Rose using the Booch methodology [5]. A "function statement" serves as a guideline for drawing the object diagram (Fig. 1b). See [4] for details on the model and class descriptions, which can also be downloaded from [6].

Typical examples of the re-use of CMSIM code are shown in the Appendix of [4]. Care has to be taken to ensure that the argument communication between Fortran subroutines is contained within the class rather than crossing to other classes, so that the encapsulation of the classes is not compromised. Most C++ compilers support cross-linking between C++ and Fortran codes, but arguments need to be passed by reference rather than value in the C++ part of the code.

The STL helped to simplify the model design: some list objects have been designed as vector attributes in appropriate classes and some member functions return STL vectors. In the main program, we extensively used STL for the implementation of variable length arrays and linked lists.

The input for the track reconstruction are the coordinates of measured hits and the measurement errors, as well as the detector geometrical information.





a) b)

c)

```
             | ----- Reconstructed track id: 3        ||              | ----- Reconstructed track id: 4
             |                                        ||              |
             | Track parameters:                      ||              | Track parameters:
             |                                        ||              |
             |  r    phi   z     theta  beta   1/Rtr  ||              |  r    phi   z     theta  beta   1/Rtr
             | (cm) (rad) (cm)  (rad)  (rad)  (1/cm)  ||              | (cm)  (rad) (cm)  (rad)  (rad)  (1/cm)
             |                                        ||              |
MC track: 1  | 7.925 1.979 0.802 1.470 -0.018 -0.005  ||  MC track: 2 | 7.925 1.656 0.246 1.540  0.003  0.001
             |                                        ||              |
hit id:120001 |    hit id:120001  dchi2:  0.654544    ||  hit id:120002 |hit id:120002  dchi2:  0.106214
hit id:110001 |    hit id:100001  dchi2:  2.08672     ||  hit id:110002 |hit id:110002  dchi2:  2.80786
hit id:100001 |    hit id:90001   dchi2:  0.654544    ||  hit id:100002 |hit id:100002  dchi2:  0.961897
hit id:90001  |    hit id:80001   dchi2:  0.558137    ||  hit id:90002  |hit id:90002   dchi2:  0.106214
hit id:80001  |    hit id:70001   dchi2:  0.488648    ||  hit id:80002  |hit id:80002   dchi2:  2.22553
hit id:70001  |    hit id:60001   dchi2:  0.654544    ||  hit id:70002  |hit id:70002   dchi2:  6.18551
hit id:60001  |    hit id:50001   dchi2:  0.953577    ||  hit id:60002  |hit id:60002   dchi2:  0.106214
hit id:50001  |    hit id:30001   dchi2:  0.935086    ||  hit id:50002  |hit id:50002   dchi2:  3.6026
hit id:40001  |    hit id:20001   dchi2:  2.21221     ||  hit id:40002  |hit id:40002   dchi2:  6.45174
hit id:30001  |    hit id:10001   dchi2:  2.67882     ||  hit id:30002  |hit id:30002   dchi2:  1.55332
hit id:20001  |                                       ||  hit id:20002  |hit id:20002   dchi2:  0.958787
hit id:10001  |                                       ||  hit id:10002  |hit id:10002   dchi2:  1.43631
               Total Chi2 =       11.9                ||                 Total             Chi2 =       26.5
                                                      ||
  --------------- MC track id: 1                      ||  --------------- MC track id: 2
corresponding  recon track: 3                         ||  corresponding recon  track: 4
  tot hits on rec: 12  correct hits: 10, wrong hits: 2 ||  tot hits on rec: 12,   correct hits: 12,  wrong hits: 0
```

Fig. 1. a) Class diagram for the OO-model showing the classes implemented in the model. b) Object diagram showing the interaction between the objects. c) Sample screen output for a two-track event (slightly edited for clarity).

Ultimately, these inputs should come from an OO database; but for now we store the CMSIM generated hits in formated ASCII files, which are then read to instantiate hit and layer (as well as MC_track) objects in the C++ main program. Each hit has an ID consisting of the layer ID number and the hit series number on that layer.

After the track reconstruction, a cleaning process removes all ghost tracks. Then a "comparator" establishes the correspondence between MC_tracks and reconstructed_tracks by comparing the hit IDs on each track. From that, the track- and point-finding efficiencies can be evaluated from a criterion of a "good" track > (e.g. ? 80% of hits on a MC_track found on a reconstructed_track). The output of OO model for a test with a two-track event is shown in Fig. 1c.

3 Summary and prospects

An object oriented model for the CMS central tracking reconstruction has been designed, coded in the C++ programming language and has undergone preliminary tests. The main features of this model are:

(i) Class design according to the OO paradigm, based on proven data concepts in HEP track reconstruction.
(ii) Re-use of Fortran subroutines from the Kalman filtering track reconstruction package in CMSIM as member function of various classes.
(iii) The STL, a powerful tool in C++ programming, has been rather extensively used in the model design, the class coding and the main program coding.

This pilot project served to gain familiarity with OO-analysis, OO-design and C++ implementation as well as to explore re-use of Fortran legacy code. At this point, we believe that we have basically achieved this primary goal, but plan to use this model as a basis for a more complete, precise and efficient OO track reconstruction program. See [6] for further details and new developments on the model.

4 Acknowledgements

We would like to thank Dr. Patrice LEBRUN for the great help in the initial C++ coding and to Dr. Lassi TUURA for the generous assistance on using the STL.

References

[1] The CMS Technical Proposal, CERN/LHCC 94-38 (1994).

[2] S. Qian, "Simultaneous Pattern Recognition and Track Fitting by Kalman Filtering Method for CMS Inner Tracker", CMS TN/96-001 (1/1996).

[3] D. R. Musser and A. Saini, "STL Tutorial and Reference Guide", Addison-Wesley (1996).

[4] I. Gaines et al., "Design and Test of an OO Model for CMS Track Reconstruction", CMS TN/96-122 version 2.2 (11/1996).

[5] G. Booch, "Object-Oriented Analysis and Design", Benjamin-Cummings (1994).

[6] Webpage: http://cmsdoc.cern.ch/~sijin/oo.html.