Design of the L3 Offline Software

F. Bruyant CERN, ECP Division, Geneva, Switzerland

Abstract: The L3 Software Base has been developed with rather conventional techniques. The paper emphasizes possibly conservative but nevertheless essential aspects of Software Engineering.

I. Introduction

The L3 Detector\(^1\) is one of the four detectors built around LEP, the Large Electron–Positron collider, at CERN. The main orientations taken for the L3 Software, from 1983 to 1988, are reviewed in Section II. Section III is dedicated to details of implementation in areas related to program structuring, data structures, database and production control. The simple minded techniques of Software Engineering effectively used are briefly mentioned in Section IV. The relative importance given to Sections II and III reflects the opinion of the author that, however sophisticated the Software tools may be, the major difficulty resides in identifying clearly the relations between the elements of a Software Base and in proposing adequate data structures to ensure the communication between the program modules. It also reflects the fact that, undoubtedly, the manpower dedicated to pure Offline Software activities, within the L3 Collaboration, was insufficient.

II. Main Orientations

II.1 L3 (very) specific constraints

The complete separation between Offline and Online activities, the absolute priority given to hardware and the intention to split the Collaboration into two independent analysis groups were, in the early 1980’s, the preferred guide lines. As a consequence, the structures of coordination for the Offline Software were not given much consideration. Nevertheless, free and motivated individuals (the author was one of them) tried to implement, year after year, a coherent Software Base, ultimately adopted by both analysis groups, and made operational thanks to the talent and effort of more and more physicists.

Presented at the International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics, Lyon, France, March 1990
II.2 Constituents of a Software Base

In HEP experiments nowadays, event simulation, database system, event reconstruction and production control are the corner stones of a Software Base.

- **Event Simulation**: To optimize the construction of the detector, to test the performance of the reconstruction algorithms, to check the trigger efficiency and to interpret correctly any results derived from the real data.

- **Database System**: Highly desirable for high precision large statistics experiments; it has to be secure, efficient, economical, functional and user friendly.

- **Event Reconstruction**: To produce accurate and complete results, for real and simulated data, and to present them in a way easy to be analyzed by the physicists.

A common requirement for simulation and reconstruction → Modularity

- **Production Control**: To record all conditions of event data handling history, such as LEP machine (or generator) parameters and detector parameters for the successive runs, precise specifications of the programs and their conditions of utilisation, performance and status of any submitted production jobs.

- **Other essential constituents**: Event generators, Interactive event display (and reconstruction), Calibration programs and Database builders.

For L3, a special effort has been made to implement in an orderly way all these elements.

II.3 Choice of Environment

Because of the trends, and also because of the participation of L3 members in the development of general CERN Software packages, the following choices were made in the early period:

- **Language**: FORTRAN 77, not debatable then!

- **Code Management**: PATCHY\(^2\), with multi-Pam input (PAchemy Master files) and the pam modules (the Patches); also very useful for Program structuring.

- **Simulation Package**: GEANT3\(^3\), with many attractive features, such as Detector description and automatized tracking, concept of ‘sensitive detectors’,
interactivity and graphics, auxiliary packages (e.g. FFREAD\textsuperscript{4}) to read free format data cards, or the histograming package HBOOK\textsuperscript{5}) and, most important, good physics (electro–magnetic, and hadronic from GHEISHA → GEANH\textsuperscript{6}).

Also, possibilities of extension for use in reconstruction context → GEANE\textsuperscript{7})

- Memory Manager : ZEBRA\textsuperscript{8}, with powerful data structures, computer independent sequential and random access I/O facilities.

Later, other packages have been adopted, such as KUIP\textsuperscript{9}), for interactive applications, HIGZ\textsuperscript{10}, an interface to the standard Graphics systems, PAW\textsuperscript{11}, for physics analysis on Workstations, CMZ\textsuperscript{12}, an interactive super–Patchy and FATMEN\textsuperscript{13}, for file and tape management.

III. Selected Details of Implementation

III.1 Program structure and program control

The L3 source code (about 400K lines) is spread in separate PAM files, classified as Steering, Basic or Utility. In addition to the 20 Pams of the Generator pool (either developed within L3 or adapted from foreign programs), about 20 other Pams are used to generate the Simulation and Reconstruction programs and the Database library. The Simulation and the Reconstruction Steering Pams (e.g. SIL3, REL3) may share the Basic Pams (e.g. XSL3, for the eXperimental Setup description) and the Utility Pams (e.g. UTL3). Each Steering Pam is equiped with Pilot Patches (for instance *SIGEL3 and *SDGEL3, in SIL3) which permit the user to build as many programs (e.g. SIGEL3 for complete event simulation and SDGEL3 for re-digitization pass without tracking).

The modularity of the programs can be controlled,

- at creation time, either vertically, through the Pilot patches, or horizontally, with Patchy options (e.g. for simulation, full tracking or use of shower libraries and, for both simulation and reconstruction, possibility to switch off the code of any detector component), and

- at execution time, at the level of the detector components and subcomponents, through data card options, either horizontally (e.g. simulation in all detectors but
the tracking chamber, reconstruction in muon chambers only), or vertically (e.g.
redigitization for all detectors, refit with same pattern recognition results).

III.2 Event data structure skeleton

The EVNT data structure, designed for the L3 event reconstruction\(^{14}\), is used
both for simulated and real data (Fig. 1). In the first case, a pseudo-DACQ bank
supports the GEANT3 data structures prepared by the simulation program; in
the second case, the linear chain of DACQ banks comes from the Data Acquisition
system. The corresponding information is cleaned-up, reformatted and stored
in raw data banks, the xyRW chain (x for the detector component, y for the
subcomponent, e.g. HBRW for the Hadron calorimeter Barrel).

![Diagram of the EVNT data structure skeleton]

**Fig. 1** The EVeNT data structure skeleton
The raw data are then converted into preprocessed data, stored in xyPP banks, usually after a first order (reversible) calibration and, when useful, with some selection. The xyPP banks contain, for every active readout channel tagged by its unique Offline Software Identifier, a number of quantities in units which can be directly interpreted by the physicists and displayed graphically. Of course, the DACQ and xyRW banks are dropped when no longer needed.

In first approximation, the study of a given detector component can be made independently on the others; any partial results which one needs to save, the so-called reconstructed objects, are stored in banks supported by a reconstruction fanout xREC assigned to the given detector component. Various levels of reconstructed objects can be introduced, the more complex ones being built from the information formerly stored in simpler ones. By convention, the xyPP banks are considered as the lowest level reconstructed objects and are appended to their respective xREC fanout.

The objects can be of two types, individual or collective. The kinds of constituents which can be used to build a given object are defined a priori, the constituents being themselves objects at a lower level in the same fanout or in another fanout. The definition of the relation between any kind of object and its potential constituents is encoded in the data part of the xREC fanout bank to which the object belongs. Each bank has a few header words followed, for the individual banks, by a number of standard words describing the object, and for the collective banks, by the standard words describing a member–object, repeated as many times as there are members. Then, for the individual banks only, at a position specified in the header, comes the section where are detailed, for each kind of constituent in turn, the identity of the constituents effectively used and their characteristics in context of the association to the current object.

Access to the constituent objects themselves is enabled through the definition of reference links in the given individual bank. Structural links to support banks containing any other relevant data can also be defined.

It may happen that, temporarily or permanently, objects of a given kind are incompatible, because they make reference to the same constituents. This is the
case when ambiguities cannot be locally resolved. When it is so, (complete) lists of compatible objects or (partial) lists of incompatible objects can be stored in LOCO or LICO banks. These banks, by convention, are appended to the link which precedes the one assigned to the corresponding objects.

The assignment of links within a given xREC fanout, the presence of LICO/LOCO banks, the logical relationship between the objects as well as their structural characteristics, such as number of standard words, number of potential kinds of constituents and their names, number of structural links, are encoded in the data words of the given fanout.

A precise definition of the standard words in the high level objects provides a way to develop independent analysis chains, to the extent where alternative algorithms can be introduced in any part of the reconstruction program without major perturbations in the overall logic of even: processing.

The rules defined for the individual detector reconstruction can be extended to the Across L3 combined detector reconstruction with the definition of another fanout bank, AREC. The objects belonging to AREC are built from objects belonging to one or more xREC fanout banks, possibly including AREC itself. The higher level objects within AREC are closely related to the entities which, at the end, are used for physics analysis. The definition of the relation between a given xREC (or AREC) and any other xREC fanout bank is encoded in the data part of the EVNT bank. As recently pointed out, the EVNT data structure, with its substructures which reproduce the same properties at different levels, can be qualified as fractal structure. Fractal, may be, but chaos not yet!

The PASS banks are defined to keep track of the current status of event processing for successive program passes. They support LOBR banks which make reference to all reconstructed objects, with their current status (dropped, temporarily deactivated ...), and which permit the user to select any objects for interactive applications or for event display.

III.3 Database and Production Control

The Database management system DBL3\textsuperscript{15}) is based on the Zebra RZ package
which permits one to store in random access files, and to retrieve, data objects identified by their pathname (UNIX-like directory structure) and by a set of Key values. The functionality and the performance of RZ have been substantially improved to handle large amounts of calibration constants; in particular, the Node/Key banks which reproduce in memory the database directory structure, for the data objects currently used, have been specially designed to optimize the disk-to-memory transfers.

DBL3 is also useful for other applications, for instance for the Production control\textsuperscript{15}: The programs are classified according to the Pilot patches prepared in the Steering Pams. Each program may have one or more modes of operation (e.g. event simulation with full tracking or with fast shower development). On a given Computer at a given Computing center (CC), a given Mode of operation of a given Program (PM) is identified uniquely by its PMCC Id. To execute the program, data cards are usually required. For the successive official Pam releases, the program installation templates and the standard sets of data cards are stored as data objects in specific directories of the database, with the corresponding PMCC Id recorded in a given key.

At Job submission time, instantiated copies of the program installation templates and of the I/O requests, which describe precisely the current program environment, are stored in other directories of the database. Any submitted production job receives a unique Job Identifier, which is encoded in the file name of the output files produced by the job (or recorded in a given key of the database objects created by the job). The Job Id is also stored in the event PASS banks. A summary of the job execution is finally stored on database, where it can be checked interactively for validation of the results.

IV. Software Engineering ... with apologies

IV.1 Formal Conventions

Subroutines and labelled Common names, with six characters, the first being the first letter of the Pam file. Variables in Common also with six characters, the last two being the first two of the Common name. Local variables with less than six characters.
Mnemonics for bank addresses (LBxxxx, where xxxx is the bank name), for positions within the supporting bank (KLxxxx), for the standard data word offsets and for the status bits.

Standard subroutine headers (purpose, arguments, called by, non trivial calls) and, in the code, structural indented comments with three or less stars.

IV.2 Structural Analogies

Whenever possible, Utility routines to avoid code duplication and to simplify the maintainance. Fanout steering routines calling analogous routines for the various detector components, wherever relevant.

IV.3 Documentation and Interactive Help Facilities

Within each Pam file, history, contributors and documentation patches including flowcharts, banks description and data cards description. Possibility to use the Help Directory structure of DBL3 but, as in Brassens’ song ‘Margot’, no one has time to write things that anyway no one would have time to read!

ACKNOWLEDGMENTS

Without the contribution of many physicists, the L3 Software Base would have never been operational. I would like to mention in particular L.Barone, J.Branson, R.Clare, N.Colino, F.Erne, H.Fesefeldt, V.Innocente, G.Massaro, R.Mount, E.Nagy, M.Pohl, I.Sheer, R.Wilhelm and my closest collaborator, S.Banerjee, whose efficiency and friendly assistance have been most appreciated.

REFERENCES

[2] PATCHY, H.Klein and J.Zoll, CERN Program Library I 400
[4] FREAD, R.Brun et al. CERN Program Library I 302
[5] HBOOK, R.Brun and D.Lienart, CERN Program Library Y 250
[7] Average Tracking and Error Propagation Package, V.Innocente et al. L3 Note # 572
[8] ZEBRA, R.Brun and J.Zoll, CERN Program Library Q 100
[10] HIGZ, R.Brun et al. CERN Program Library Q 120
[16] The L3 Production Control, F.Bruyant and N.Colino, L3 Note # 701