ARCHITECTURE OF THE LEP APPLICATION SOFTWARE

Jean-Pierre Koutchouk
CERN
CH-1211 Geneva 23

Abstract

A thorough analysis of the requirements for application software is presently being done for LEP. To complement and unify this approach, a synthetic or generic model of the LEP control has been developed. The accelerator is viewed as an automaton with a limited number of macroscopic states. Each state is represented by a data structure. The transition between states is actuated by tasks (application programs); they are grouped into families with similar functionalities with respect to the machine states. It is shown how this organization attempts to map the requirements of reliability, flexibility and extendability into the LEP control system.

1 INTRODUCTION

A significant effort is presently invested in the specification of the application programs necessary to operate LEP. This paper describes an architecture integrating them in a consistent entity. The first aim is to enforce systematics in the operation of LEP which should result in an improved efficiency, reliability and flexibility . Several of the principles proved their value in the field when operating the CERN-ISR. The second aim is to define at an early stage the data structures which interface the application programs.

2 OPERATION REQUIREMENTS

There are basically three modes of operation, each defining some specific requirements :

1. The commissionning/start-up/maintenance mode requires easy access to individual hardware or data modules and the ability to quickly write small programs. This is best handled following the principles of the SPS control system [1].
2. The machine studies require mainly flexibility :
   - the ability to invoke on-line the LEP models to modify important parameters, in a controlled way.
   - an efficient storage and retrieval of measured data, properly tagged with whatever information is necessary to interpret them.
   - the logging of the machine evolution to help understanding the observations.
3. The luminosity production requires efficiency and reliability . These are achieved not only by the quality of the software but even more by the minimization of the consequences of hardware/software/human errors. Indeed, together with the limited capabilities of predicting the luminosity and background, these errors are responsible for the "inertia" in changing parameters related to the significant learning period.

In order to act on all these aspects, it is necessary to reduce complexity, to have fault-tolerant operation and to facilitate recovery :

- operation should be modular at a higher level than a program: LEP operation can be viewed as an automaton making transitions between a small number of important states; these states are reflected in data structures, that may be pointed at, archived...
- Three instances of the machine states should be considered simultaneously : what a state should be (Reference), what it is (Current) and what it would be (Target) if calculated corrections would be applied.This allows a systematic separation of computations from control and provides a wide spectrum for checks and recoveries.

3 CLASSIFICATION OF APPLICATION PROGRAMS

In order to fulfill the operation requirements, the data must systematically be separated from the programs. On figure 1 the data are grouped so as to emphasize three classes of application programs which have different roles.

- the real-time applications programs directly interact with the LEP hardware or with the data structures describing the machine states. They must fulfill the requirements of efficiency and reliability.
- the model programs are accelerator physics programs which predict accelerator and beam parameters (Reference states). They must be general and expandable to provide the required flexibility. It is intented for LEP to use MAD [2] for the optics.
- the study programs allow the progress of accelerator physics applied to LEP. This is usually referred as modelling. They need not conform to any standard; they can only interact with some well-defined data-structures to avoid any interference with the real-time processes.

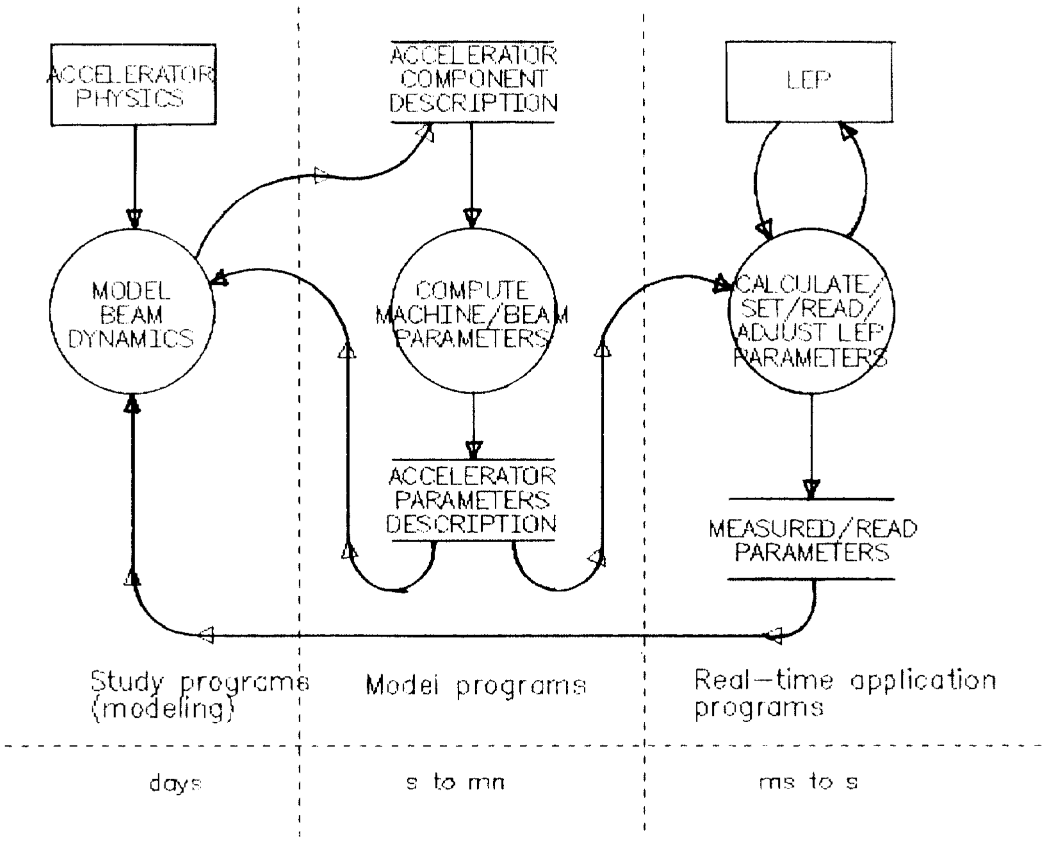
ACCELERATOR PHYSICS
ACCELERATOR COMPONENT DESCRIPTION
LEP
MODEL BEAM DYNAMICS
COMPUTE MACHINE/BEAM PARAMETERS
CALCULATE/ SET/READ/ ADJUST LEP PARAMETERS
ACCELERATOR PARAMETERS DESCRIPTION
MEASURED/READ PARAMETERS
Study programs (modeling)
Model programs
Real-time application programs
days
s to mn
ms to s

Figure 1: classification of application programs

The purpose of this classification is economy : it shows that only a subset of the application programs - the real-time ones - require a strong organization and hence an added complexity in the implementation. There is more freedom for the other programs. Unity is guaranteed by the interface to common data structures and probably by a standardization of the man-machine interface.

4 ARCHITECTURE OF THE REAL-TIME APPLICATION PROGRAMS

The architecture shown on figure 2 embodies the major operation phases and the data structures necessary to fulfill the requirements defined so far.

1. In the data preparation phase, the Reference sequence of states is either created or reloaded from the archives; the LEP model is invoked for Twiss parameter calculation,... which become part of the states. The resulting data structure is called the Reference Data Set. It remains unchanged unless a learning mechanism is activated.

2. The hardware control function is dedicated to settings/readings and measurements; its inputs are found in the Reference Data Set in case of stable operation. Its outputs define, at least partially, the actual machine states. They are stored in the Current Data Set. The information unit being a state, any measurement is bound to a description of the accelerator; this ensures that it can be correctly interpreted. The Current Data Set furthermore keeps track of state modifications.

   Check of the transition rules is done at this stage to prevent errors.

3. The "feedback" loop processes information from the Current Data Set and produces target states (e.g. tune shift) stored in the Target Data Set. The target states not only consist in increments to controllable parameters but equally in predictions of beam or machine parameters.

   The hardware control function finds then its input in the Target Data Set. Facilities are foreseen to select a specific state in the Target Data Set, display its corresponding predictions, scale the increments by a factor or carry knob-driven control.

4. The learning mechanism allows modifications of the Reference Data Set, for example when better performances are achieved in one run; this is especially interesting for machine studies.

   If a definite improvement is achieved, it may be archived for use by later runs. In this case, the preparation phase is reduced to the reloading of the Reference Data Set, or even to nothing in case of stable operation.

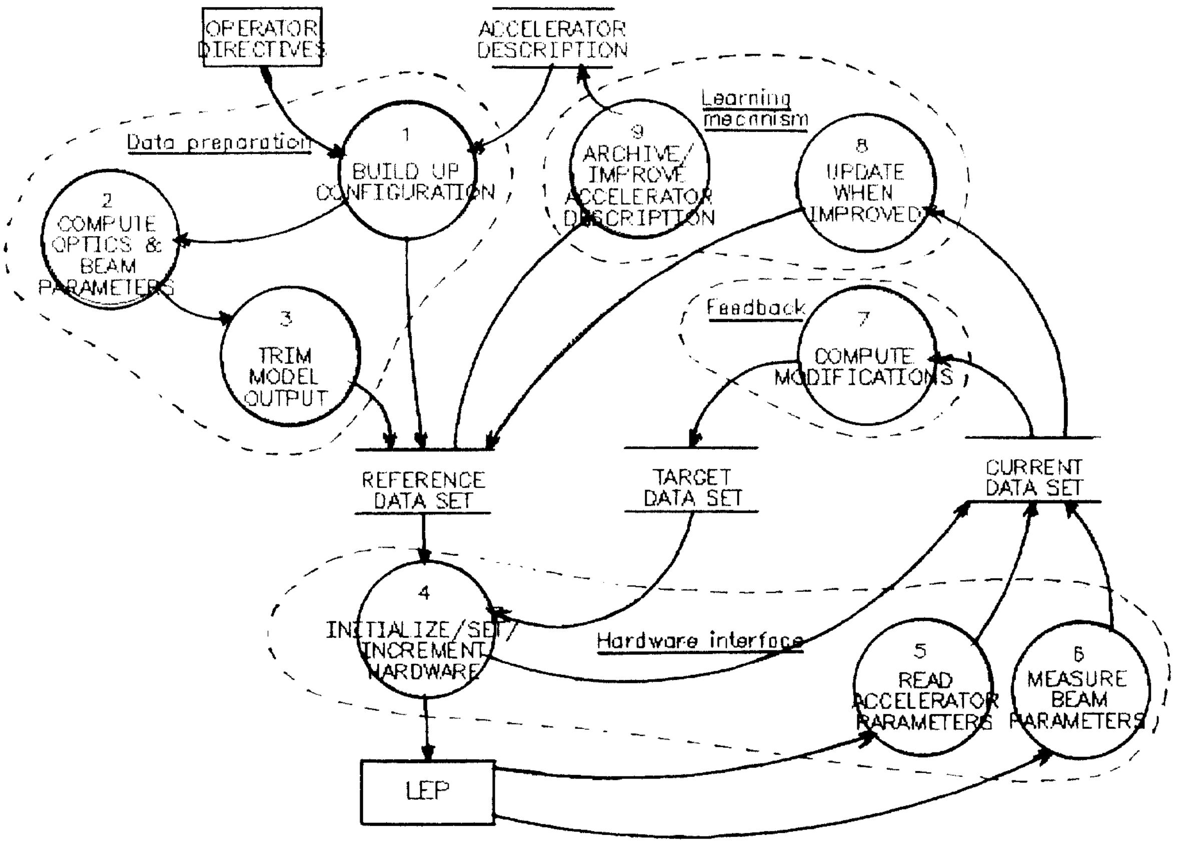

Figure 2: architecture of application programs

5. A sophisticated access mechanism to the archives is being developed [3], based on associative search, in order to fully exploit the accumulated information and to allow the management of this large volume of data.

The concepts developed should make it easy to trace the development of a run, allow to go back in the sequence of states or state modifications, or even to restart from the Reference Data Set. Operation efficiency and flexibility should thus be improved.

5 CONCLUSION

This architecture of the LEP application software has given a framework for the analysis of the individual tasks and has helped in the definition of the required data structures. The modularity introduced should equally allow a smooth transition from manual to automatic operation.

6 ACKNOWLEDGEMENTS

I would like to thank my colleagues of the LEP Application Analysis working group and especially J. Poole for their comments.

References

[1] M.C. Crowley-Milling, CERN 75-20, Dec. 1975

[2] E. Keil, F.C. Iselin CERN/LEP-TH/85-15, 1985

[3] J. Poole, CERN-LEP Note 571, 1987