SOFTWARE DEVELOPMENT FOR THE SSC

J. A. Appel, C. Day, D. Linglin, S. Loken, P. LeBrun, E. May,
M. Shapiro, and W. Zajc

Report of the Subcommittee on Offline Computing Requirements
for the Workshop on Triggering Requirements
for High Energy/High Luminosity Hadron-Hadron Colliders

Fermilab
November 11-14, 1985

1 Introduction

This note deals with proposed software development methodologies for SSC experiments. While these techniques are being discussed under the auspices of the Offline Computing and Networking group, it is essential to realize that such methods are best applied in a global fashion, so that they may also be applied to the on line software efforts, as well as to software being developed for Monte Carlo and design purposes.

2 Structured Analysis and Structured Design

The sub-group agreed that structured system design is required to write and maintain the very large, interrelated software structures necessary for successful operation of an SSC detector. This section briefly describes structured analysis/structured design (SASD), what advantages may be expected from its usage, and what reservations were expressed concerning the same.

Very roughly, SASD applied to system development may be compared to "top-down" methods of program development. That is, one begins with a high-level specification of the desired system performance. This description is continually expanded and refined by identifying the forms of the input and output data, and the operations performed upon them. "Data" in this context is very

general. For example, it could consist of actual detector event data, alignment parameters, Monte Carlo output, accelerator control signals, etc.

The expansion of the system into its sub-levels proceeds via a fairly formal procedure using such tools as Data Flow Diagrams and a Data Dictionary. What at first glance appears to be a large rigid overhead in fact becomes a valuable self-documenting feature. Thus, two immediate benefits of SASD are to provide a detailed study and model for the entire system *before coding begins*, and to provide documentation for the same.

Currently, ALEPH is the only HEP experiment that has made a "full" commitment to this approach. Approximately 50 physicists and programmers have taken a one week course in SASD given by a commercial vendor (Yourdon, Inc.), at a cost of approximately $1000 per person. The DØ collaboration has also begun applying SASD to the design of its total software system, although at a less formal level of commitment than the ALEPH collaboration.

Costs of SASD are largely those of the commercial training seminars. Software tools are also available at small cost ($5000) to maintain the hierachical diagrams and dictionaries that provide the system description. It is expected that these relatively small expenses are completely outweighed by the benefits of increased programmer productivity, ease of modification, and availability of documentation inherent to SASD products (see also Section 4 on manpower estimates). Associated with the complete system description is the ability to rapidly introduce new participants to the details of the software structure, a non-trivial feature of an experimental effort that will span a decade.

The intricacy of the complete system description was seen as a partial liability to some members of the committee who were concerned that the extreme

formality of this approach may appear stifling to the traditional physicist/programmer. This is an undeniable psychological/sociological phenomena, although one that probably applies to all aspects of SSC experiments rather than only the software effort. Associated with this were questions of management of the total software endeavor. Will high level software managers simply be administrators, delegating tasks along the tree-like organization chart corresponding to the system description? In this case, the lowest level will correspond to the stultifying encoding of totally specified program units. This is the converse of "traditional" HEP software efforts, and may present conceptual difficulties to all involved.

Finally, it should be noted that SASD is intended to proceed as far as possible without specifying the language of implementation. To avoid inadvertently designing in "pure Fortan," it may be necessary to have as members of the design team programmers whose languages are not limited to Fortran. Similarly, the system design should be as independent of the actual hardware configuration as possible. At some point this must break down (e.g., with the actual read-out, or with a vector processor), but it is in fact a laudable design goal to envision processes as operations on data that are independent of the actual machine.

3 Tools and Environments

Closely associated with software development are the tools provided for code and file management. Many experiments have used PATCHY as a code management device, but it is not clear that it is really a code management *system*. The distinction lies in the automatic logging of edit changes and file access, which are not provided by PATCHY. In this sense, PATCHY should be regarded as a meta-language for compilation time editing and code transport, rather than a product capable of forcing the tracking of edit histories. These shortcomings

are at least partly responsible for the decision by CERN to use HISTORIAN as a future code management system. A similar product (CMS) is available from DEC, which has been used very successfully by TPC and currently by CDF. The major shortcoming of this tool being its lack of transportability to other machines.

Also discussed were various software products that have been used in high energy physics. Clearly, there are tools that are specific to high energy physics (e.g., GEANT). However, there are other utilities where this is not so clear. For instance, it is commonly assumed that HBOOK is a package uniquely tailored to the needs of physicists analyzing large amounts of data. To what extent is this assumption true? Are there commercial products available with the same or more advanced features? A second example is HYDRA. Is HYDRA a valuable tool for describing linked data structures, or is it a patchwork job designed to provide this service for a language (FORTRAN) that does not support these concepts? Two major problems identified with the purchase of commercial products were the high cost (if sold on per processor basis, esp. for a farm!) and the lack of commitment to maintenance, extensions, new operating systems, etc.

Finally, the usage of available data base management systems remains largely unexplored by high energy physics. It is clear that producing and maintaining large blocks of structured data (calibration constants, pedestals, alignment measurements) is a problem ideally suited to such techniques. Further study and/or consultation with industrial users of data bases is required.

4 Time and Manpower Estimates

Here we present a simple estimate of the total software effort required for a large SSC experiment. We will use 500K as a working number for the total number of lines of code. Given that the UA1 offline code alone

consisted of roughly 400K lines, this is almost certainly an underestimate. Typical programmer productivities for *debugged* lines of code are of order 4 lines/day, or 1000 lines/yr. This estimate is consistent with industrial averages, in fact it may be an overestimate based on TPC experience (coding exclusively on one family of machines!). These numbers lead to a *minimum* time estimate of 500 man-years, or 100 FTE's for 5 years, corresponding to a real cost of $50 million dollars for software development.

Capital costs for this development are small in comparison (other than the actual hardware on which the software will eventually run). We assume that every programmer is provided with a terminal supporting full screen editing, interactive debugging, and local graphics. Also essential are electronic mail (e.g., through HEPNET), and good quality text processing (e.g., Massll or TEX). Desirable but not essential at the per programmer level are workstations and local hard copy devices, although it should be clear that individual productivity decreases as these resources become scarce.

Given the high cost of developing software, it is not unreasonable to ask how much of this code may be shared among experiments. Clearly, much of the code developed for a given HEP application has been borrowed from previous experiments. That is, the algorithms are very often the same (clustering, track-fitting, vertex-finding, etc.). It is often the details of the implementation, (the data format, calling sequences, common block structures) that prevent re-use of code. Although there is the very real danger of propagating errors between otherwise independent experiments, this is an area that deserves further investigation.

5 Recommendation

There is a clear need to evaluate the effectiveness of particular SASD methods in designing large HEP software efforts. A test case would provide a

valuable service to the community, by exposing physicists from different experiments to these techniques and by identifying those areas of difficulty specific to our needs. We propose that a working group be formed to apply SASD to the creation of a detailed Monte Carlo design study of a reference detector. The benefit of this is obvious, in that the end product will be not only an *in situ* evaluation of SASD, but also an essential piece of software for future physics input to SSC detectors.