LBL-30191

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

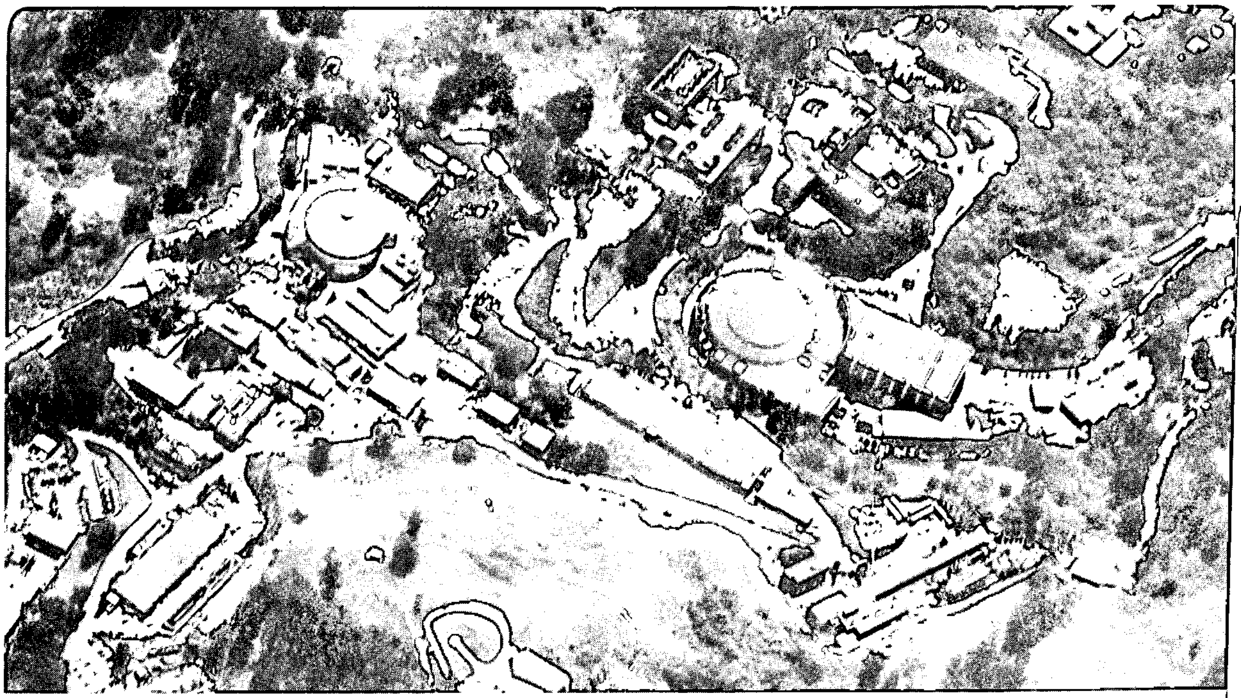
Information and Computing Sciences Division

Presented at the Computing for High Luminosity and High Intensity Facilities Conference, Santa Fe, NM, April 10–13, 1990, and to be published in the Proceedings

Implementation and Management: A Summary

S.C. Loken

January 1991



Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098.

IMPLEMENTATION and MANAGEMENT
A Summary*

S. C. Loken
Lawrence Berkeley Laboratory
University of California, Berkeley, CA 94720

ABSTRACT

We review the sessions related to Implementation and Management. The papers presented at the Conference cover a wide range of important issues in software engineering and management. They indicate a trend toward more use of commercial systems and standards. This trend will likely have a significant influence on plans for future systems.

INTRODUCTION

In any review that covers many parallel and plenary sessions, we must omit important contributions. I have tried to select those areas that seem to indicate new directions for our community and that may be of interest to us when the next conference in this series is held in Tsukuba next year.

The issues that I will address here are the following:

- Management
- Software Engineering
- Architecture
- Database Management
- Human Interfaces
- Planning for the Future

In preparing this summary, I have ommitted important contributions in other areas and I apologize to those who have been left out.

*This work was supported by the Director, Office of Energy Research, U. S. Department of Energy under Contract DE-AC03-76SF00098.

MANAGEMENT

The recent experiences at LEP in bringing on large computer systems and software can teach us a lot about how to manage large system. Based on the experience of developing the ALEPH Online system, von Ruden [1] suggested a set of guidelines for success.

- Learn management skills
- Assemble a good team
- Simulate the overall system
- Design an integrated hardware/software system
- Minimize the number of subsystems
- Enforce Software Engineering rules
- Don't build computers
- Avoid politics

It is not easy to follow all these guidelines, especially the last, but it is clear that our systems are becoming larger and more complicated. We need to improve the level of management on computing projects and we need to become more aware of the need to develop an integrated system. It will be difficult to enforce engineering rules across the project and for this reason, we need to look at system architectures that will isolate people who do not follow the rules. An approach to this is described in the section on architecture below.

SOFTWARE ENGINEERING

In past meetings, the discussions on software engineering seemed to focus on one methodology, Structured Analysis/Structured Design (SA/SD). It is becoming clear that this is only one possible part of the solution to the problem of engineering computing systems. The full solution will require coding rules, code management and distribution, and many other pieces.

It is interesting, however, to review the success that projects have had using SA/SD. It has been applied to a number of experiments including ALEPH[2], D0[3], ZEUS[4] and Fenice[5]. The results have been mixed but, in all cases, the groups agree that the methodology did improve the software design. In all cases, however, the lack of tools with which to maintain the analysis and design was a major problem. It was not possible to update the documentation easily and, as a result, it became obsolete. The SA/SD methodology was not followed through the coding phase.

An important aspect of the analysis/design efforts was the modeling of data using the Entity Relationship Model[6] or the ALEPH DAta MOdel (ADAMO)[7]. These tools provide a way to represent the complex relationships between data objects. Data modeling is an increasingly important part of the general problem of Database Management.

DATABASE MANAGEMENT

It has become increasingly clear that database management is an important component of High Energy Physics software. The problems will be even more severe for SSC and LHC experiments with more complex detectors and more events to deal with.

The use of commercial Relational Database Management Systems and of home-grown packages was reviewed by Rimmer[8]. Commercial systems have been used successfully by the LEP builders and by some of the LEP experiments. These systems do satisfy, at least, some of the requirements of the machines and of experiments. There are, however, features that are needed and are not provided by the commercial systems and many groups have developed packages to provide the needed functions. Many of these have been built using the ZEBRA/RZ packages.

The optimal use of database systems requires the use of data modeling techniques to describe the objects in the database and the relationships among the objects. The Entity-relationship Model[6] has been very successful in providing such a description. Some experiments have used this model as part of SA/SD or other software engineering. The ALEPH group has integrated data-modeling into their software design and has developed the ALEPH DAta MOdel (ADAMO)[7]. The ZEUS experiment has also adopted ADAMO[4].

The use of database systems and data modeling will continue to increase as the data management challenge increases. Groups need to experiment with these techniques and get experience with different products. To avoid becoming dependent on specific products, it is useful to adopt a high-level standard such as the Standardized Query Language (SQL).

ARCHITECTURE

Architecture seems to be an over-worked phrase these days. In our context we mean a software structure that allows us to build robust programs. It has been clear for some time that we need to design systems that have small modules that work, and can be tested, independently. This approach has been demonstrated to improve programmer productivity and reduce software maintenance costs.

One technology for building modular systems that interoperate is called the software bus[9]. There are now a number of commercial implementations of this technology and there is increasing interest in defining a standard for a software bus. A pilot project using one bus standard was described by Grieman[10]. While this approach requires more effort in the implementation phase, the result, so far, has been modules that are easier to maintain and to reuse in other applications.

A formal approach to the development of software modules is Object-oriented Programming. While this approach is not yet wide-spread in High Energy Physics, some pilot projects show considerable promise. The paper by Kunz[11] provides an introduction to Object-oriented Programming techniques. These techniques have been used very successfully in the Reason Project described in the next section.

HUMAN INTERFACES

The development of good user-interfaces is critical to improving the quality of scientific software. The popularity of graphical interfaces in the personal computer market demonstrates the need to follow this example in the development of programs for High Energy Physics. The Reason Project[12] has developed an interactive data-analysis package that utilizes the interface-building software that is supplied in the NeXT computer. The program does not yet have full functionality but it demonstrates the great potential for this approach. The ease with which they have developed the program indicates the benefits of using high-level software tools to develop scientific software.

Another new software product that is likely to have a significant impact is the Application Visualization System (AVS) from Stardent Computers. Using AVS, the physicist can develop sophisticated graphics applications without writing code. The D0 experiment has been using AVS for 3-D event displays[13]. The approach shows great promise and will be used to develop more general tools.

PLANNING FOR THE FUTURE

The lifetime of experiment is now significantly longer than the career of a graduate student or post-doc. We need to recognize this in planning experiments. We cannot depend on students or post-docs to maintain code for the life of the experiment. We must design software that is reliable and can be maintained by others. The use of software engineering tools for design, documentation and testing will be critical.

The present generation of experiments already spans more than one cycle in computing technology. This introduces new challenges for the software developer. For the SSC and LHC era, the problems will be even greater. Experiments must plan from the beginning to evolve as the technology changes.

To ensure that systems will survive changes in technology, new software should be developed to utilize commercial products to the greatest extent possible. All software, whether developed or purchased, should adhere to standards. The industry will then assist in moving critical software to new technologies.

We are already seeing standards having a significant impact of computing in High Energy Physics. The use of UNIX is widespread, both for high-end workstations and as a source of cheap computing cycles. The Internet protocols (TCP/IP) which were virtually unknown a few years ago are now an important part of the network traffic. X-windows and new graphics standards (GKS and PHIGS) are gaining acceptance. This evolution will surely continue. The use of standards will ensure that investments in software will maintain their value as technology changes.

CONCLUSIONS

We are beginning to see a major change in the way that physicists do their computing. The availability of high-level interfaces and other commercial products will have a very significant impact of future projects. As with any new tool or technique, we need to get experience with the new software technologies.

REFERENCES

References

[1] W. von Ruden. Managing a Large Data Acquisition System - Can Success be Programmed? These Proceedings.

[2] G. Kellner. Develpoment of Software for ALEPH using Structured Techniques. In P. Kunz and T. Schalk, editors, *Computing in High Energy Physics: Proceedings of the International Conference on Computing in High Energy Physics, Asilomar, 2-6 February 1987*, page 229, 1987.

[3] J.J. Linnemann et al. The Use of SA/SD Methods in D0 Doftware Development. In P. Kunz and T. Schalk, editors, *Computing in High Energy Physics: Proceedings of the International Conference on Computing in High Energy Physics, Asilomar, 2-6 February 1987*, page 245, 1987.

[4] R. Loveless et al. ZEUS Hardware Control System. In R. C. E. Devenish and T. Daniels, editors, *Proceedings of the International Conference on Computing in High Energy Physics, Oxford, England, 10-14 April 1989*, page 313, 1989.

[5] A. Antonelli et al. SA/SD and Entity-Relationship SA/SD Techniques in the Fenice Experiment. These Proceedings.

[6] P. P. Chen. ACM Transactions on Database Systems, 1 9, 1976.

[7] S. M. Fisher and P. Palazzi. Using a data model from software design to data analysis: what have we learned? In R. C. E. Devenish and T. Daniels, editors, *Proceedings of the International Conference on Computing in High Energy Physics, Oxford, England, 10-14 April 1989*, page 169, 1987.

[8] E. M. Rimmer. newblock Databases for Large Detector Systems - Experiences at LEP and Future Needs. These Proceedings.

[9] D. E. Hall et al. The Software Bus: A Vision for Scientific Software Development. In R. C. E. Devenish and T. Daniels, editors, *Proceedings of the International Conference on Computing in High Energy Physics, Oxford, England, 10-14 April 1989*, page 211, 1989.

[10] W. Grieman et al. Experience Using a Software Bus to Build Reuseable Scientific Software. These Proceedings.

[11] P. Kunz. An Introduction to Object-oriented Programming Techniques for High Energy Physics. These Proceedings.

[12] B. Atwood et al. The Reason Project (Life without FORTRAN) These Proceedings.

[13] T. G. Trippe. Private communication.