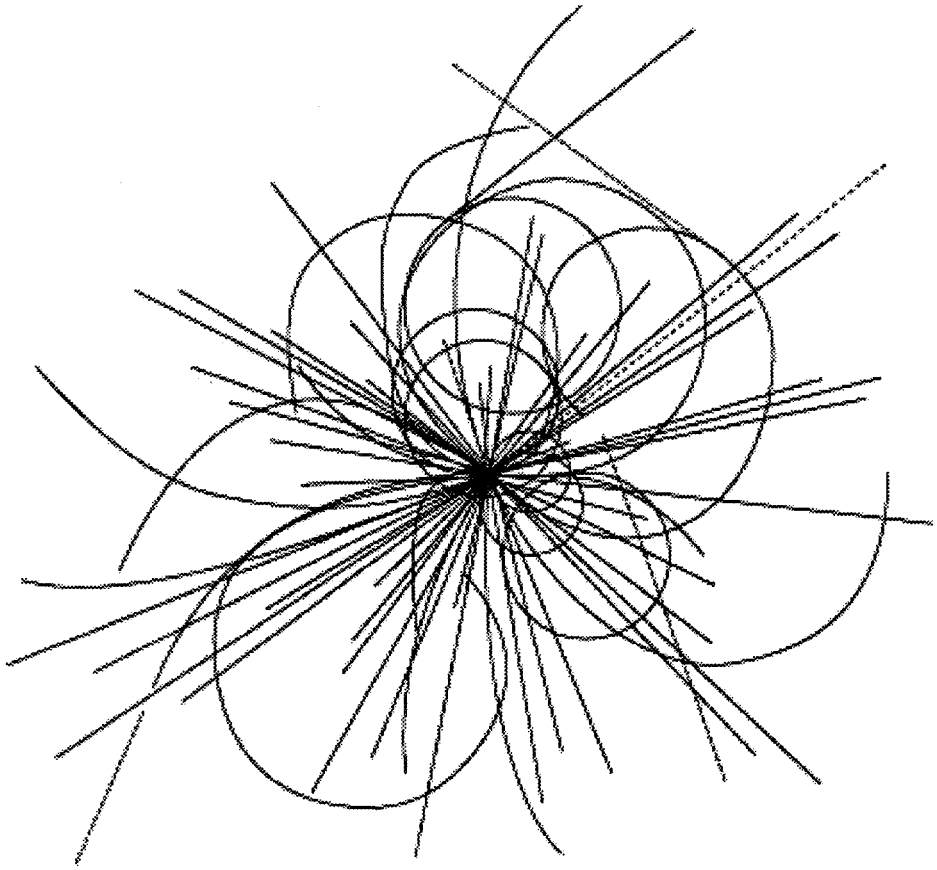
Conf-9310317--1

SSCL-Preprint-547

SSCL-Preprint-547
December 1993
Distribution Category: 400

J. Huang

Bridging Fortran to Object Oriented Paradigm for HEP Data Modeling Task



Superconducting Super Collider Laboratory

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

SSCL-Preprint-547

Bridging Fortran to Object Oriented Paradigm for HEP Data Modeling Task*

J. Huang

Superconducting Super Collider Laboratory†
2550 Beckleymeade Ave.
Dallas, TX 75237, USA

December 1993

*Presented at the Third International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics, October 4–8, 1993, Oberammergau, Germany.

†Operated by the Universities Research Association, Inc., for the U.S. Department of Energy under Contract No. DE-AC35-89ER40486.

THIRD INTERNATIONAL WORKSHOP ON

SOFTWARE ENGINEERING, ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS FOR HIGH ENERGY AND NUCLEAR PHYSICS

October 4 - 8, 1993

OBERAMMERGAU, Oberbayern, Germany

Under the Sponsorship of the

GERMAN PHYSICAL SOCIETY and

EUROPEAN PHYSICAL SOCIETY

BRIDGING FORTRAN TO OBJECT ORIENTED PARADIGM FOR HEP DATA MODELING TASK

Jean Huang
Superconducting Super Collider Laboratory,
2550 Beckleymeade Avenue,
Dallas, Texas 75237, USA
jhuang@ssc.gov

ABSTRACT

Object oriented (OO) technology appears to offer tangible benefits to the high energy physics (HEP) community. Yet, many physicists view this newest software development trend with much reservation and reluctance. Facing the reality of having to support the existing physics applications, which are written in FORTRAN, the software engineers in the Computer Engineering Group of the Physics Research Division at the Superconducting Super Collider Laboratory have accepted the challenge of mixing an old language with the new technology. This paper describes the experience and the techniques devised to fit FORTRAN into the OO paradigm (OOP).

1. Background

Traditionally, physicists who design and implement particle physics experiment software use the FORTRAN language. While FORTRAN is well-suited to writing algorithms, its homogeneous elements of ARRAY and COMMON BLOCK are very inadequate for complex data structure design. To compensate for the data-structuring deficiency in the FORTRAN language, the high energy physics (HEP) community has developed several solutions of their own. ZEBRA[1], the hierarchical structure; YBOS[2], the list of banks; and ADAMO[3], the relational model, are among the most commonly used data-management systems.

Since the 1980s, different approaches have been taken to implement the complex algorithm and data abstraction for physics application software. Instead of hanging onto the FORTRAN language with its constraint of old technology, groups of physicists have adapted the newest trend in the computer industry, the object oriented (OO) approach. The PION project[4] at CERN, has used Object Oriented Paradigm (OOP) techniques for a specialized graphics application; the SLAC Reason project[5,6], has implemented a graphical interactive analysis environment; and recently the Gismo project[7,8,9], originally developed on a NeXT computer using Objective-C, utilized OOP techniques. These projects were not only successful in testing the OO technology, their works are having a significant influence in changing the way physics applications are designed and developed.

From the computing point-of-view, it is inevitable to seriously consider using object oriented technology if the tangible benefits offered by the computer industry are to be realized. The physics

applications map quite well to the object oriented paradigm, since the original concepts are of an abstract nature; however, technology migration of any sort is a gradual process. It takes years to complete such a transition. Many physicists will continue to devote their energy to analyze HEP data using the Fortran language and will view this newest software development trend with reservation and reluctance. Facing the reality of having to support FORTRAN applications, the physicists and software engineers in the Computer Engineering Group (CEG) of the Physics Research Division (PRD) at the Superconducting Super Collider Laboratory (SSCL) have accepted the challenge and have done some experiments that mix the old language with the new technology.

The following sections describe various approaches to bridge the language to the object oriented paradigm, as well as lessons learned from the experiments. The experience will be applied to the data-modeling project, the objectives of which are to provide a data-modeling environment and to establish a standardized data model and its repository systems for supporting physics computing activities.

2. Generic Interface Approach

The first experiment done by the PRD computer engineering group was the Object Class Library for the SZ++ Package by Dr. B. Traversat. This package is an initial C++ version for hierarchical data structure memory management services that was originally provided by the ZEBRA Management System. It organized ZEBRA banks into a double-linked list class, and further defined bank attributes through derived classes. The initial work started with six classes. Figure 1 shows the inheritance of the classes. The following list is a brief description of each class:

Double-linked List Base Class implements a generic double-linked list.

Zebra Bank Base Class derived from the dblink. It adds basic ZEBRA bank information.

bank_head Class derived from the bank class. It serves as a link point.

bank_fix Class derived from the bank class. It implements a fixed-size bank.

bank_var Class derived from the bank class. It implements variable-size banks.

SZ Class manages the entire event structure and implements the SZ API to Fortran.

The organization of the data bank is separated from the design of the application programming interface. This approach leads to the design of a platform that spares the user from worrying about the underlying details of data-management systems. The following are the application interface subroutines of the SZ++ package for the Fortran language:

szbini() initializes the SZ control structure.

szfill() fills a bank.

szget() gets bank value.

szdrop() drops a bank.

szsurv() displays structure of a bank.

szshow() displays content of a bank.

szexist() tests whether a bank tree name exists.

sznum() returns the max bank number.

szwipe() wipes out a bank.

Although the SZ++ package provides a generic Fortran interface to the object oriented realm, due to its close relationship with the ZEBRA package, the design is very much bank-oriented.

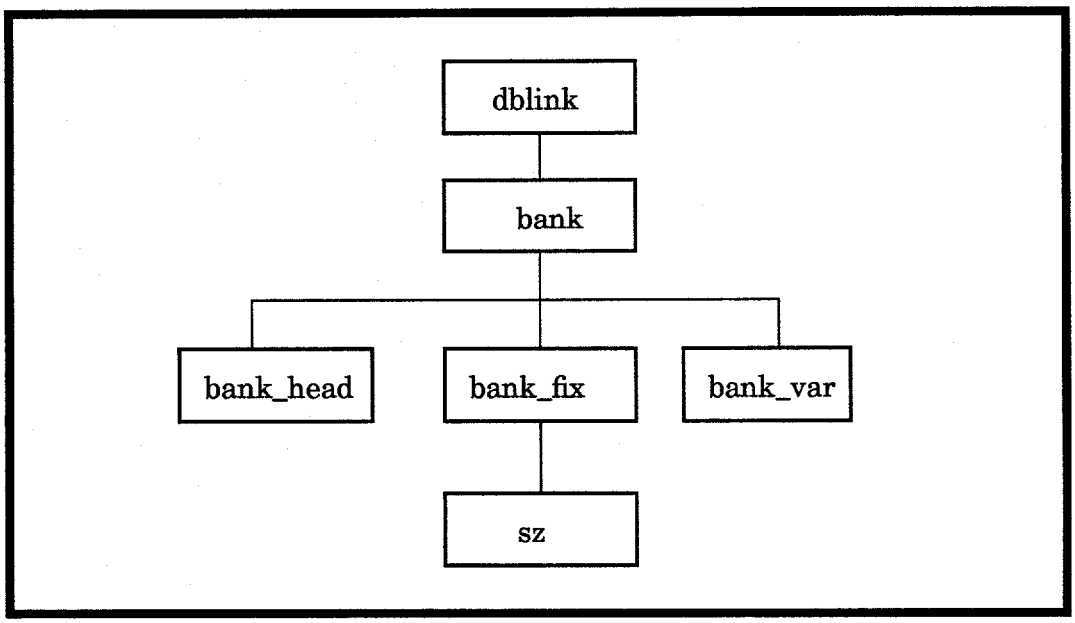


Figure 1: SZ++ Classes

3. Ad Hoc Approach

There are several projects in PRD that involve programs written in mixed languages of Fortran, C, and C++. They come in two different forms: the C or C++ main program calls Fortran subroutines, or vice versa. These projects are usually concentrated on the database and graphical user interface designed for storing and retrieving data with a limited set of functionality.

These projects were able to take advantage of the most updated computer technology of the databases and graphical user interface, but none of them put much consideration into providing a unified platform that would allow Fortran applications to work freely. Most of the time, a small modification in the header file structure requires changes to be made in the Fortran subroutine to reflect the mapping. This volatility renders impossible large-scale projects that require services for the general applications.

Figure 2 shows the call pattern of the ad hoc approach.

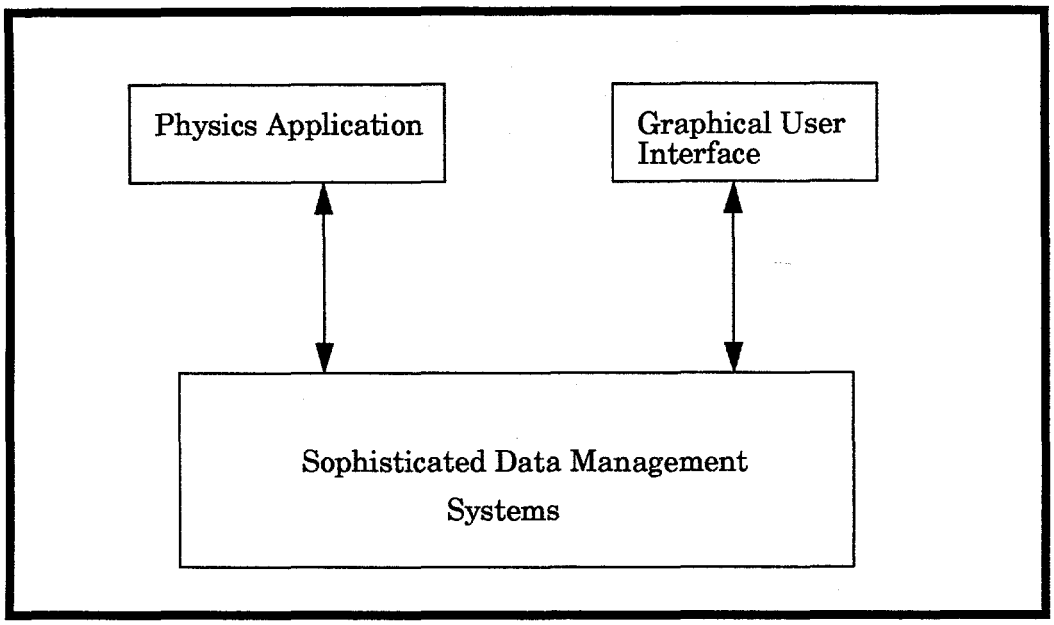


Figure 2: Ad Hoc Approach

4. Bridging Concerns

When bridging the Fortran language to the OO paradigm, the major concern is how to communicate with the more sophisticated data-structuring facility of a OO-suitable language. Both generic and ad hoc approaches encounter problems related to naming of routines, parameter passing and call positioning:

- ***Naming***

This problem is usually solved by adding an underscore to the subroutine names on the Fortran side.

- ***Calling***

The concern is which language should be used for writing the main program, and its associated linking considerations.

- ***Parameter passing***

This deals with the matching between the scattered parameters of the Fortran and the attributes defined in a more sophisticated language.

All the issues stated above are compiler-dependent. Porting software between systems may require some attention to the differences between compilers. According to the experience gained in experimental projects done in the Physics Research Computing Department (PRCD), it is not difficult to mix the Fortran language with C++ or C.

5. Conclusion

Lessons learned from these OO experiments will be applied to the development tasks of the joint Data Modeling (DM) Project effort of the Gamma, Electrons, and Muons (GEM) Collaboration, Solenoidal Detector Collaboration (SDC), and PRCD. The purpose of the DM Project is to develop the standardized data-modeling environment and HEP data models necessary to support computing activities of GEM and SDC.

Since mixing the programming languages is not a difficult task, the future DM effort will focus on the HEP data-model design and a generic platform that provides data-access services to software developers. Eventually, this data-modeling environment, which also includes tools for data analysis, data design, and data definition, will be integrated with the SSCL Framework[10] Object Management Services environment and the Physics Research Integrated Computing Development Environment (PRIDE)[11].

With the increasing complexity and advancing pace of computer technology, it is obvious that the tools and services provided by the DM environment will be acquired mostly from the computer industry, research, and academic institutions. Evidently, object oriented technology continues to gain momentum in setting the direction of today's computer software development. In order to take advantage of the technology offered by industry, the object oriented approach has been chosen for development work of the Data Modeling project.

The C++ language will most likely be used due to its close relationship with the OO technology and its wide acceptance by the software community. The development of computer technology in the physics community, however, is perceived in different shades of light. Changing from one programming language to another in a community with diversified awareness is a slow process. To best serve the GEM/SDC collaboration, an access mechanism that supports mapping of multiple programming languages, especially the Fortran language, is necessary until the majority has made the transition.

6. References

1. R. Brun, et al., *ZEBRA — Kernel Data Structure Management System*, CERN Computer Center Program Library Long Write-Up, Q100.
2. David Quarrie, et al., *YBOS programmers reference manual*, Fermilab, CDF Note No. 156.
3. S.M. Fisher, *The ADAMO data system*, CERN/ECP/PT xx/yy Real..., November 1991.
4. J. Bettels and D. Myers, *The PIONS Graphics System*, IEEE *Computer Graphics and Applications*, Vol. 6, No. 7 (July 1986).
5. W.B. Atwood, et al., *The Reason Project*, SLAC-PUB-5242 (April 1990) 11p.
6. Paul F. Kunz, *Physics analysis tools*, SLAC-PUB-5520 (April 1990) 11p.
7. W.B. Atwood, et al., *GISMO: An object oriented program for high-energy physics event simulation and reconstruction*, CERN Preprint ECP 91-15.
8. W.B. Atwood, et al., *Gismo: C++ Classes for HEP, C++ Report*, March-April 1993.
9. W.B. Atwood, et al., *The Gismo Project — C++ at work in high energy physics.*
10. S. Frederiksen, et al., *Framework Software Project Assignment Plan* and *Framework Software Project Execution Plan*, SSCL, PRCD R40-000010.

11. J. Burton, et al., *Physics Research Integrated Development Environment (PRIDE)*, presented at the Third International Workshop on Software Engineering, Artificial Intelligence and Expert Systems for High Energy and Nuclear Physics, October 1993.