EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

CERN/DRDC 94-50
DRDC / P59 (rev)
14 December 1994

A PERSISTENT OBJECT MANAGER FOR HEP

revised version of CERN/DRDC 94-30

Ryszard Zybert
School of Physics and Space Research
University of Birmingham

Pavel Binko, Gunter Folger, Jamie Shiers (spokesman)
CERN/CN-ASD

Otto Schaile
Freiburg University

Predrag Buncic
GSI Darmstadt

Boris Klokov
IHEP, P.O.Box 35, Protvino, Moscow Region

Aneta Baran, Antoni Cyz, Andrzej Sobala, Witold Wajda
Institute of Nuclear Physics
Krakow

Vincenzo Innocente
INFN, Sezione di Napoli, Napoli, Italy

Giovanni Organtini
Universita' di Roma "La Sapienza" and Istituto Nazionale di Fisica Nucleare - Sez. di Roma.
P.le Aldo Moro, 2 - 00185 ROMA (Italy)

Sunanda Banerjee
Tata Institute of Fundamental Research, Bombay

Nobuhiko Katayama
Laboratory of Nuclear Study, Wilson Lab,
Cornell University,
Ithaca, New York, 14853, U.S.A

Bill Grieman, Doug Olson, Craig Tull
Lawrence Berkeley Laboratory, Berkeley, USA

21494

Abstract

We propose to perform research in the area of a Persistent Object Manager for HEP. Objects are typically created by a process and cease to exist when that process terminates. Such objects are called Transient objects. Persistent objects, on the other hand, are those which continue to exist upon termination of their creating process and may be accessed by other processes. It is expected that any system based upon this research will work primarily but not necessarily exclusively in an Object Oriented environment. Target applications include follow on or replacement products for existing packages such as GEANT, HEPDB, FATMEN, HBOOK, experiment specific code and, most importantly, event data. Strong emphasis will be placed on the use of standards and/or existing solutions where-ever possible.

Friends of P59

Ed May, David Malon
Argonne National Laboratory

Chris Walters
CALTECH

Paolo Calafiura, Claudio Bruschini, John Apostolakis, Sven Ravndal, Simone Giani, Maarten Ballintijn, Sergio Santiago
CERN/CN/ASD

Helge Meinhard, Boris Khomenko, Chris Onions, Nick Sinanis
CERN/ECP

Adam Para, Krzysztof Genser
FERMILAB, Batavia, IL 60555, USA

Tim Bell
IBM

Viatcheslav Ivanovich Klyukhin
IHEP,
Protvino, Moscow region, RU-142284, Russia

Robert Grossman
Laboratory for Advanced Computing
University of Illinois at Chicago

Piotr Malecki
Institute of Nuclear Physics,
Dept. of High Energy,
Krakow, Poland

Jean Pierre Vialle
LAPP Annecy

Kors Bos
NIKHEF, Amsterdam, The Netherlands

Christian Arnault
Orsay

Steve Fisher
Rutherford Appleton Laboratory

Andres Sandoval
GSI, Darmstadt

Nick van Eijndhoven
Subatomic Physics Dept.
Utrecht University
P.O. Box 80.000
NL-3508 TA Utrecht
The Netherlands

1 Introduction

This document proposes the creation of a DRDC project to investigate the issues concerning the storage and manipulation of a persistent objects for LHC era HEP experiments.

Objects are typically created by a process and cease to exist when that process terminates. Such objects are called Transient objects. Persistent objects, on the other hand, are those which continue to exist upon termination of their creating process and may be accessed by other processes.

Existing DRDC projects (RD13, RD41, RD44) are committed to the exploration of *Object Oriented* (OO) techniques in a wide variety of key areas for HEP computing in the LHC era. All of these projects will need to handle persistent objects but none of them are addressing this question directly. This document therefore proposes an investigation into the question of storage and manipulation of persistent object data for the LHC era.

We start by explaining why a persistent object manager is required, we describe an architectural model in which it could be developed and how it is related to the standards developed by the Object Management Group (OMG) and the Object Database Management Group (ODMG) that currently dominate the domain. A potential design is described and we indicate how it can benefit from the work of other HEP projects. We propose a set of milestones for the project and outline the necessary resources in terms of manpower and funding to complete this proposed plan of work.

2 Why we need a persistent object manager

In High Energy Physics, we typically deal with things such as histograms, calibration data, detector geometry descriptions, production control information, meta-data concerning collections of events and, of course, event data itself. In today's environment, we use various *data structures* to represent the above information and *data structure managers* to handle these data structures both in memory and on persistent storage. In an Object Oriented environment, these data structures and the algorithms that act on them would be replaced by *objects.* Although modern programming languages provide many powerful facilities for managing and manipulating objects, an area that is typically weak is that of *object persistence.* In the commercial world, solutions such as *Object Oriented Databases* are emerging which combine object oriented features with traditional database facilities, including the provision of persistence. An important consideration regarding the HEP environment is that of scale. The exact data volumes that will be recorded at the LHC are still uncertain, but event data in the Petabyte range (10^{15} bytes) and calibration data in the 100 GB range (10^{11} bytes) is currently foreseen. This is dramatically different to existing domains to which OO techniques have been applied and is expected to have implications on the suitability of commercial solutions and on the design and implementation of any HEP specific solution.

Nevertheless, commercial products are now starting to offer many of the features that are required for HEP persistent storage systems and it is important that we understand their advantages and limitations. Current research in other HEP software projects indicates that a suitable architecture is appearing based on emerging "standards" (often de-facto, rather than de-jure), but that the components themselves may not be capable of supporting HEP's requirements in terms of scale and/or performance. We do not believe that the requirements of other scientific or commercial fields will approach those of HEP in the LHC era and are therefore sceptical that a complete commercial solution will be available in an appropriate timeframe. Nevertheless, we strongly emphasize the need to adopt the interfaces outlined by the various standards so that we can take full advantage of standard components for a persistent object data manager (PODM) should they exist or come to market between now and the running period of LHC.

3 Why the system should not be fully specified at this stage

As there are many unknowns concerning the computing environment that will exist in the LHC era, care should be taken to avoid over-specifying the system at this early stage. As a result, we concentrate more on the architecture than the implementation.

Below we list some of the features that we believe a persistent object manager should offer. This list should not be considered as final - an important component of the first phase of the project will be to define more completely the exact requirements, in close collaboration with the LHC experiments.

4 Why OO

As with most new technologies, Object Orientation is seen as a panacea for all the ills that have besieged software systems for decades. Being more objective, we can expect OO to give our software the following attributes:

- Faster development
  If systems are mainly built from existing objects (classes) then projects can be completed in a much shorter time-scale.
- Better Quality
  Both the enforcement of modularity through encapsulation and the reuse of already proven classes tend to improve the quality of a system.
- Flexibility and maintainability
  The modularity provided by the OO approach allows the design of systems to be modified in a controlled manner, both in terms of changes to the underlying processes and information and in terms of the selection and linking of objects to carry out a given function.
- More realistic modeling
  The use of objects in the analysis stage tends to provide a closer model of the way we naturally think of systems.

5 Towards the Persistent Object Data Manager (PODM)

5.1 Requirements

Below is a list of general requirements that should be satisfied by a persistent object management system. We hope that some (if not all) of the requirements can be filled by adopting techniques or products from other projects or industry. The list has been driven by the needs of LHC experiments and with the intention that a number of successive implementations may be required as the system evolves. It must be stressed that this requirements list is not exhaustive and will be refined, with the help of the collaborators from various experiments, during the first phase of the project.

- Independent of operating system and machine architecture
- Independent of underlying storage technology
- Provide means for data definition, manipulation and query
- Effective limitless capacity for data size and number
- Provide interfaces for the most popular programming languages to all objects (irrespective of how they were defined)
- Support for concurrent access to data from multiple applications
- Capable of being used in a distributed environment (e.g over a network of machines)
- Provide graphical tools for data manipulation and query without programming
- Capable of exploiting parallel file systems
- Capable of processing persistent data written by a previous version of the software.

5.2 Architectural Model

As the number of applications and vendors of OO technology has increased it has become clear that some form of standardisation is necessary if the advantages of the technology are not to be lost in the chaos caused by such divergence. The Object Management Group (OMG) appears to be succeeding as a single emerging standards body for OO technology, although clearly this may not continue to be the case throughout the entire lifetime of the LHC experiments.

5.3 Object Management Group (OMG)

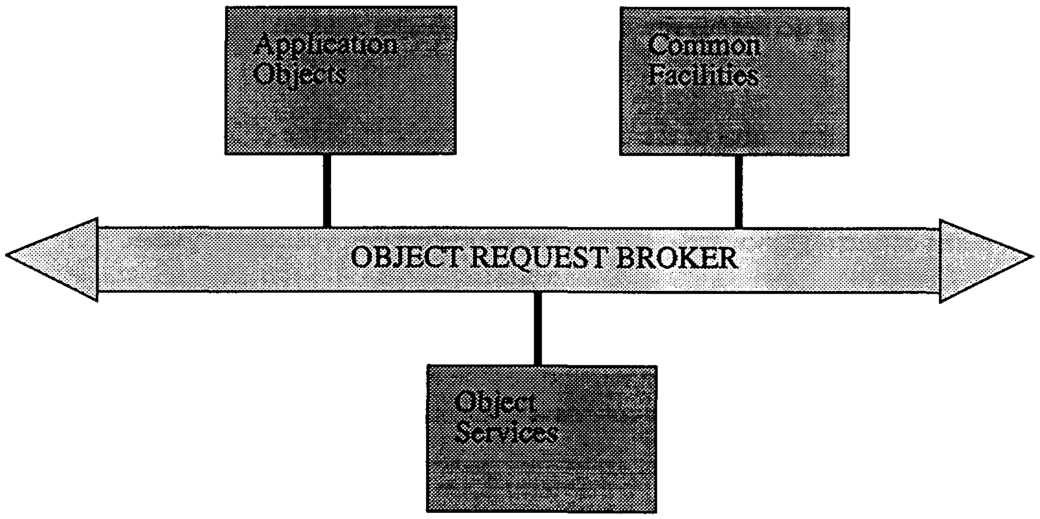
The Object Management Group (OMG) is an international software industry consortium with two primary aims:

- promotion of the object-oriented approach to software engineering in general, and
- development of command models and a common interface for the development and use of large-scale distributed applications (open distributed processing) using object-oriented methodology.

In 1990 the OMG published its Object Management Architecture (OMA) Guide document. This document outlines a single terminology for object-oriented languages, systems, databases and application frameworks; an abstract framework for object-oriented systems; a set of both technical and architectural goals; and an architecture (reference model) for distributed applications using object-oriented techniques. To fill out this reference model, four areas of standardization have been identified:

1. Object Request Broker, or key communications element, for handling distribution of messages between application objects in a highly inter-operable manner;
2. Object Services, is a collection of services with object interfaces that provides basic functions for realizing and maintaining objects.
3. Common Facilities: a collection of interfaces and objects that provide general purpose capabilities useful in many applications.
4. Application Objects: those that are specific to particular end-user applications.

Figure 1 OMG's Object Management Architecture Overview



5.3.1 Common Object Request Broker Architecture (CORBA)

The OMG publication entitled 'Common Object Request Broker: Architecture and Specification'. more commonly known as simply CORBA, provides the mechanisms by which objects transparently make requests and receive responses, as defined by OMG's ORB. The ORB provides interoperability between applications on different machines in heterogeneous distributed environments and seamlessly interconnects multiple object systems. Dynamic Invocation Interfaces (DII) permit clients to dynamically compose requests and invoke them.

5.3.2 Interface Definition Language (IDL)

IDL is OMG's language for specifying the interface to objects. An IDL compiler front-end has been made freely available, courtesy of Sun Microsystems. This release of the Interface Definition Language (IDL) implementation is divided into three parts:

- A main program for driving the compilation process
- A parser and attendant utilities
- One or more back ends (BEs) for taking the processed input and producing output in a target language and target format

5.3.3 CERN membership of OMG

CERN's application to become an associate member has recently been accepted by OMG which will provide us with immediate access to all the publications and the right to participate to subgroups.

5.4 Persistent storage techniques

With traditional development techniques it has been necessary to code explicit I/O commands to retrieve and store data. Such systems are normally based on one of the following techniques:

- conventional file
  Data are stored in to files using the operating system's file system. This technique is suitable for single-user applications that do not require concurrent multi-user data access and is widely used in HEP software packages.
- relational database
  Data are mapped onto the structures of the underlying database in terms of tables and rows. The data base provides support for multi-user concurrent access and a specific language for data definition and manipulation (SQL). Such databases offer more functionality than flat files but have more overheads in terms of performance and demands on the underlying operating system.

In OO systems retrieval and storage can be transparent, as they can be built in to the methods of the classes. Thus whenever a method for an object is invoked it can first check whether the object is in memory and fetch it if not. Similarly, if a method updates the attributes of an object it can ensure that the updated object is stored correctly. In this way developers can treat all objects as if they were permanently in memory.

5.5 Object Oriented Database Management Systems (ODBMS)

Object-Oriented Databases are databases that support objects and classes. They are different from the more traditional relational databases because they allow structured sub-objects, each object having its own identity or object-id (as opposed to a purely value-oriented approach). In addition, they provide support for object-oriented features such as methods and inheritance. It is also possible to provide relational operations on an object-oriented database. ODBMS allow all the benefits of object-orientation, as well as the ability to have a strong equivalence with object-oriented programs, an equivalence that would be lost if an alternative were chosen, such as a purely relational database. Rather than providing only a high-level language such as SQL for data manipulation, an ODBMS transparently integrates database capabilities with the application programming language. This transparency makes it unnecessary to learn a separate data manipulation language (DML), obviates the need to explicitly copy and translate data between database and programming language representations, and supports substantial performance advantages through data caching in applications.

5.5.1 Object Database Management Group (ODMG)

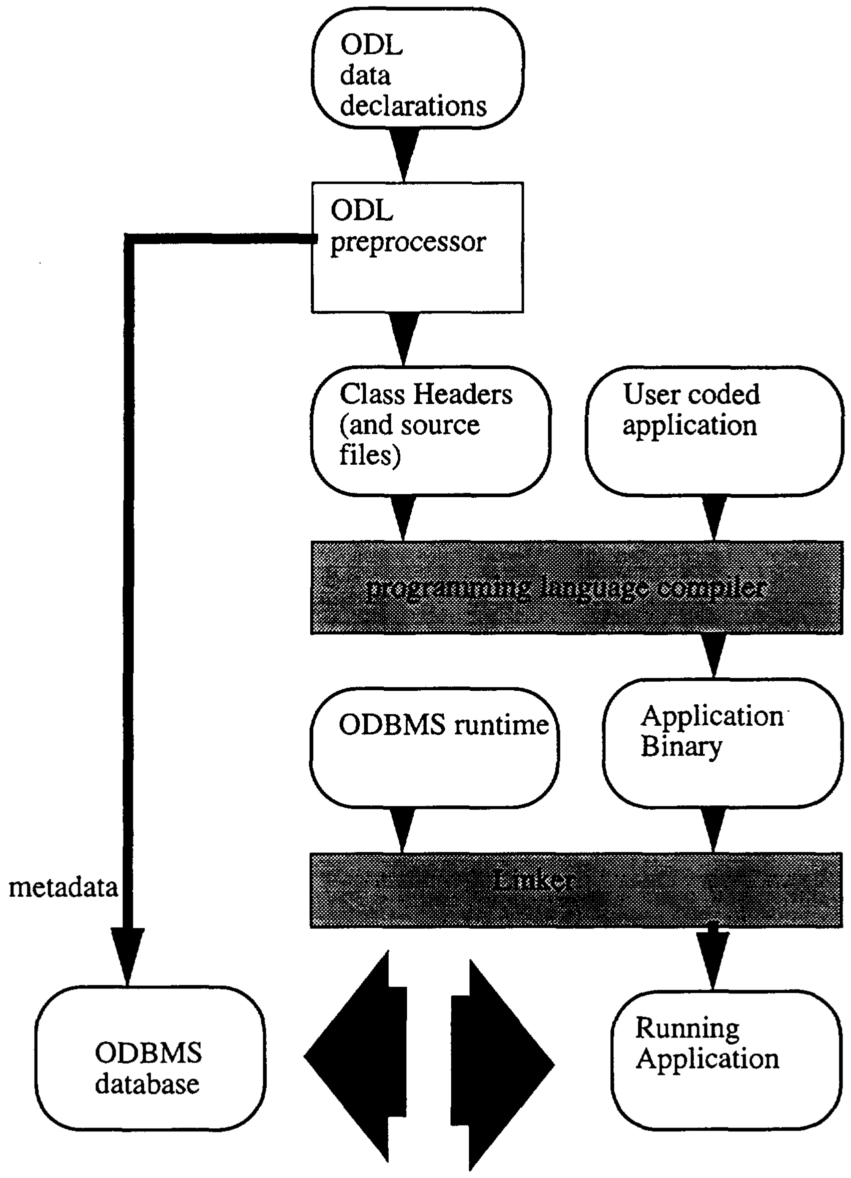
The ODMG is an informal consortium of ODBMS vendors working on standards to allow portability of customer software across their products. By supporting the same interface, they expect to significantly accelerate progress towards effective adoption of the standard. By submitting it to relevant organizations (OMG, ANSI, ISO, STEP, PCTE, CFI, etc.), they plan to encourage commonality among the object database portion of all those standards.

To date, the lack of a standard for ODBMS has been a limitation to their more widespread use. Much of the success of relational database systems is due to their standardisation in terms of a data model and SQL language. Standardisation is even more important for ODBMS because their scope is more far-reaching, integrating the programming language and database system, and encompassing all of an application's operations and data. A standard is critical to making such applications practical.

The ODMG specification consists of the following:

- Object Model
  The common data model to be supported by the ODBMS that integrates with OMG's Object Model. Components (such as relationships) that are necessary for ODBMS have been added to the OMG model.
- Object Definition Language (ODL)
  The data definition language for the database. ODL is a superset of the OMG's Interface Definition Language (IDL) and provides a programming language independent mechanism to express user object models (schemas)
- Object Query Language (OQL)
  A declarative (non-procedural) language for querying and updating database objects for interactive and programmatic query as an extension of SQL
- Programming language bindings
  Define how applications written in the supported programming languages (C++ and Smalltalk for the moment) can manipulate persistent objects including object definition, manipulation, and query.

Figure 2 ODMG hierarchy of languages and steps in the generation of an ODBMS application



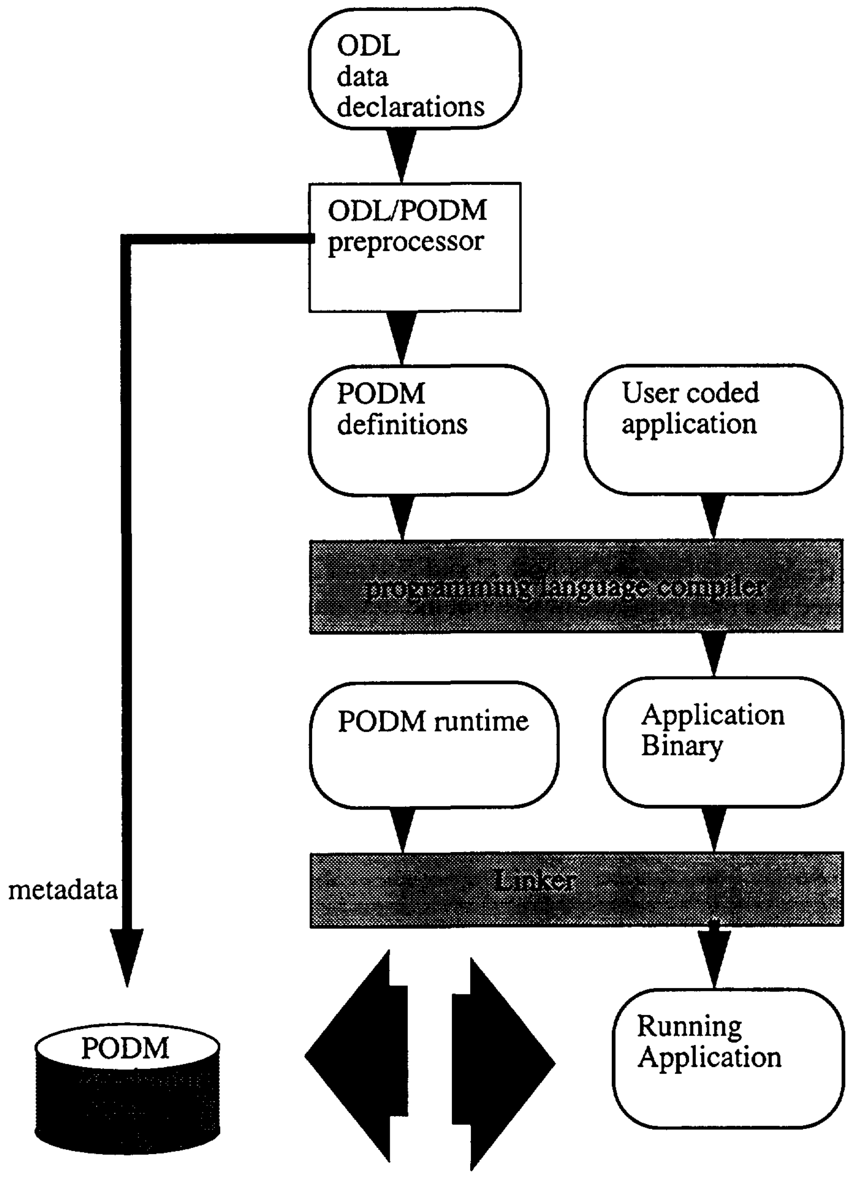
5.6 ODBMS alternatives

Applications that do not need the full power of an ODBMS can save in both complexity and execution overhead by developing a way to store and retrieve their own objects. ODBMS have not yet developed to a state that can fully support LHC's requirements in terms of performance and capacity and may not do so in the required timescale. As a safe guard against such a situation, it is proposed that we investigate the development of an alternative storage mechanism specific to HEP but which conforms to the OMG architecture model in such a way that data can be migrated to new storage mechanisms as they become available. A means of transferring data from existing HEP storage mechanisms (e.g. ZEBRA) to this new environment should be envisaged.

This alternative storage mechanism could have the following components:

- A ODL parser that converts the data definition into a PODM representation
- A class library for OO languages (initially C++) to perform object manipulation
- An Application Programming Interface (API) for non-OO languages (FORTRAN and C) to perform object manipulation

Figure 3 A Possible Design for PODM within the ODMG architecture framework



5.7 Related projects

Many HEP experiments have addressed the issue of data persistence by developing their own storage techniques. We have surveyed the most well known of these packages and intend to use the knowledge gained from their implementations as a basis for the PODM development.

- PASS

An essential component that was identified early on in the PASS project [1] was that of a persistent object manager. The PASS project wrote their own system, named Ptool [2], and also worked with a commercial solution, ObjectStore [3] from ObjectDesign Inc., which is available for numerous platforms. Much of the work performed in the PASS project is directly relevant to this one, although PASS are explicitly targeting only event data, and are placing much more emphasis on the questions of mass storage than we currently intend to do. However , we must clearly benefit from their experience and establish appropriate links with members of the PASS team. In particular, we wish to gain from their experience with CORBA [6] and CORBA compliant systems.

- NA49

NA49 is a fixed target heavy ion experiment at CERN SPS. As part of the experiment's software, a package for the manipulation and storage of data, called DSPACK[5], has been developed. The DSPACK system, introduces object oriented concepts that are available to the C or Fortran programmer. DSPACK supports the following OO concepts: polymorphism, methods and inheritance. In true OO style, the appropriate method is invoked by sending a message to an object. DSPACK may be used with or without ZEBRA and can certainly be considered as an interesting prototype for any future system. By offering OO features in both C and Fortran environments, it allows the benefits of OO to be demonstrated in a real experiment without the trauma of a complete change of programming language.

5.7.1 Related DRDC projects

The work outlined in this proposal is related to several other DRDC projects. We intend to collaborate closely with these projects in order to profit from and contribute to the work that has been already been done.

- RD13

The RD13 project was approved in April 1991 for the development of a scalable data taking system suitable to host various LHC style data acquisition studies. As part of this work they have investigated the use of commercial ODBMS and software development using OO methods and CASE tools.

- MOOSE

Moose will study the viability of the Object Oriented approach for software development for High Energy Physics reconstruction and analysis code at the LHC. By working on some well defined prototypes experience will be gained in Object Oriented analysis, design and implementation. They have started adding object persistence to the prototypes using a home-made file based scheme for Eiffel programs (MOP [25]) and are evaluating several OO analysis and design methods.

- GEANT4

This project will be a first attempt to re-design a major package of CERN software for an object-oriented environment and intends to contribute to the production of a general framework for the LHC off-line software. GEANT4 will be an important application for the PODM development.

6 Project Phases

6.1 Phase 1

The first phase of the project will be devoted to obtaining a more complete requirements list and would involve detailed discussions with the LHC collaborations and with experts from both inside and outside CERN. It would also include evaluations of existing packages such as Ptool and ObjectStore. A limited amount of prototyping will be necessary but will be restricted to simple cases, with the goal of identifying areas requiring detailed prototyping in a second phase. This first phase will last for some six months.

6.2 Phase 2

The goals of the second phase of the project will be to undertake detailed prototyping, to be judged by a set of metrics to be developed, and which will enable a detailed design document to be written. It is not possible to precisely specify the areas of investigation but current thinking would include the following items of work. Although event data is clearly the most important area to target, current thinking suggests that we will start with some of the simpler applications, such as a production database, which will help us gain experience with both OO and object persistence.

- Evaluate the ODMG's Object Definition Language (ODL) as a suitable data definition language. This evaluation will take the form of a prototype which will add persistence to a chosen application.
- Evaluate possibilities of language independence by storing objects in one language (e.g. C++) and retrieving them in another (e.g.Eiffel).
- Develop a back-end to one of the publicly available OMG's Interface Definition Language (IDL- a subset of ODL) which will implement persistence using ZEBRA. This step is intended to show a possible migration path for existing applications towards the new OO system. Alternative back-ends can be developed for other HEP specific persistent systems such as MOP and PASS.
- Develop metrics for each of these application areas by which prototypes may be judged.
- Develop a prototype application using an ODBMS that is conformant to the ODMG 93 standard as a means of evaluating the functionality and performance of ODBMS.
- Investigate distributed object management by implementing an application using a commercial CORBA-compliant object request broker.
- Add ODMG Object Query Language capabilities. Evaluate if OQL is suitable for HEP applications

6.3 Phase 3

The third phase of the project would be the implementation phase. That is, the results of the first two phases, in particular the various prototypes, would be used as input to the design and implementation of a HEP persistent object manager.

6.4 Proposed milestones for the first year

We propose the following milestones from the above list of work items as those which should be completed in the first year.

- Evaluate the ODMG's Object Definition Language (ODL) as a suitable data definition language for HEP.
- Develop a prototype application , such as a calibration/geometry/production database, using an ODBMS that is conformant to the ODMG 93 standard, e.g. O2, as a means of evaluating the functionality and performance of ODBMS.
- Investigate distributed object management by implementing an application using a commercial CORBA-compliant object request broker, such as Orbix or ORBeline from PostModern.

6.5 Workshops

It is our intention to organise a number of workshops in which PODM issues can be discussed between P59 authors, friends and others who are also active in this area. The workshops are being organised to coincide with popular HEP conferences and activities in order to minimize travel. A preliminary list of workshops has been schedules as follows:

- February-March 1995 at CERN
- September 1995 at CHEP'95

7 Manpower

7.1 Members and Friends

P59 collaborators are grouped into "members" and "friends". "Members" are those people who have signed the current proposal and will make some significant contribution to the project itself.

"Friends" are people who support the project and participate to the level of reading and commenting on any documents or proposals that are produced.

- Phases 1 and 2: 30-36 man months per year. (Includes one full-time associate).

- Phase 3: It is too early to predict exactly how much manpower will be required for this phase of the project, but experience with previous (simpler!) systems would suggest that a minimum of 24 man months per year would be required.

8 Resources

Our financial requests to the DRDC amount to 100 KSF per year. In particular, the following resources are requested:

- Funds for trial licenses of appropriate software (e.g. ObjectStore): 50 KSF per year.
- Funds for hardware on which software can be developed and/or evaluated: 40KSF in the first year.
- Funds to pay for travel and subsistence for external P59 collaborators: 100 KSF per year (this does not include the associate in 1995.)

9 Relationship between P59 and Mass Storage

Clearly, the most demanding application that must be solved in terms of Object Persistence is that of event data. The ATLAS and CMS technical proposals predict a requirement to store multiple Petabytes (PB) of new event data per year. This data cannot all be stored on disk and so clearly an interface to some Mass Storage System is required. At this stage of the project, when it is completely open as to whether the data will be stored in files or some other format, it is not possible to specify our Mass Storage requirements much beyond the scale required, namely for a multi-PB system. The majority of the work that is carried out in the area of Mass Storage Systems is to some degree connected to the IEEE Computer Society Reference Model for Mass Storage Systems, in the sense the model compliance is claimed or simply that the work is presented at one of the regular IEEE Mass Storage Symposia. As the spokesman for this project is also heavily involved in the various IEEE Mass Storage activities (member of the technical committee, member of organising committees of symposia from 92-96 etc.), no problems in terms of information flow are expected in this area.

10 References

Below is the list of other documents referenced in this proposal. A number of books have also been used as a general basis for the philosophy behind this paper but are directly referenced in the paper "Object Oriented Report, Geoff Seel, Financial Object Resources Ltd., Cambridge Market Intelligence Ltd., ISBN 1-897977-33-6."

[1] Petabyte Access Storage Solutions. The PASS Project Architectural Model. Proceedings of CHEP94 (to be published).

[2] Ptool. The Design and Evaluation of a High Performance Object Store.Laboratory for Advanced Computing, University of Illinois at Chicago, March1994.

[3] Object Design Inc., Burlington, MA. ObjectStore Reference manual.

[4] The ZEBRA Data Structure Management Package. Overview of the ZEBRA System, CERN Program Library Q100.

[5] DSPACK Object Oriented Data Manager. DSPACK tutorial CERN Program Library Q125.

[6] Object Management Group. The Common Object Request Broker: Architecture and Specification, Revision 1.1, OMG TC Document 91.12.1, 1991.

[7] Object Management Group. Persistent Object Service Specification, Revision 1.0, OMG Document numbers 94-1-1 and 94-10-7.

[8] The Object Database Standard, ODMG-93, Edited by R.G.G.Cattell, ISBN 1-55860-302-6, Morgan Kaufmann (publishers).

[9] C++ IOStreams Handbook, Steve Teale, ISBN 0-201-59641-5, Adisson Wesley.

[10] Object-Oriented Databases, Bindu R. Rao, ISBN 0-07-051279-5, McGraw Hill.

[11] Pattern Class and Persistence for Large Projects, Jiri Soukup, ISBN 0-201-52826-6, Addison Wesley.

[12] GEANT detector simulation package, Long write-up, CERN Program Library W5013.

[13] FATMEN Distributed File and Tape Management System, Long write-up, CERN Program Library Q123.

[14] HEPDB database management system, Long write-up, CERN Program Library Q180.

[15] HBOOK Statistical analysis and histogramming package, Long write-up, CERN Program Library Y250.

[16] The Physics Analysis Workstation, Long write-up, CERN Program Library Q121.

[17] The BOS dynamic memory management system, DESY Internal report R1-99-01, DESY, Germany.

[18] The Cheetah Data Management System, Proceedings of the 14th INFN Eloisatron Project: Data Structures for Particle Physics, Erice, Italy, 1990.

[19] The HYDRA data structure package, CERN TC Program Library.

[20] Jazelle - an Enhanced Data Management System for High Energy Physics, Proceedings of CHEP 90, Santa Fe, New Mexico.

[21] The YBOS memory management system, Programmers Reference Manual, CDF Computing Group, Fermilab.

[22] The ZBOOK dynamic storage management system, Long write-up, CERN Program Library Q210.

[23] RD41 (MOOSE), An Object Oriented Software R&D project.

[24] DRDC proposal P58, GEANT 4: an Object Oriented toolkit for the simulation in HEP.

[25] MOOSE Object Persistence, http://www.cern.ch/OORD/io.html

11 Acknowledgements

We gratefully acknowledge the help of Bob Jones, CERN/ECP, in preparing this revised version of the proposal.