DØ Note 2819

DØ Run-II Computing Language and Event Data Persistency Recommendations

December 19, 1995

Language and Data Structure Working Group

D. Adams, M. Diesburg, S. Fuess, B. Gibbard, H. Greenlee, J. Hoftun, J. Linnemann
B. May, L. Paterno, M. Paterno, S. Protopopescu, R. Raja, S. Snyder, J. Womersley

This Working Group was charged with recommending what programming language or languages should be used by DØ for off-line software in Run II and to recommend a Zebra replacement for use in the handling of data, especially event data, in conjunction with that language recommendation. A list of high level goals for the software development effort to be based on these recommendation follows.

1. The required software must be complete and functioning correctly on a schedule appropriate to Run II.

2. The quality of the software should be high. It should be efficient. It should be easy to debug, maintain and evolve. It should be "user friendly", easy to understand and convenient to use.

3. The software should be as portable as possible. It should operate on as wide a range of platforms and operating systems as practical and should require as little change as possible in moving from one platform or operating system to another. In particular, it should be possible to move the data transparently from one platform to another.

4. It should be possible for the definition of the data associated with particular objects to evolve with time without serious disruption. In particular, there must be a level of backward compatibility which allows access to previous versions of data in an automatic manner.

5. Production of the software should require as little technical effort as possible. The number of individuals and the amount of time they spend doing software, as opposed to physics, should be minimized.

6. The software should support the migration of algorithmic modules across as wide a range of activities within DØ as possible. It should specifically allow modules developed for reconstruction to function with little or no modification in the Level 3 filtering environment, the on-line monitoring environment and the high level analysis environment.

The majority of the Working Group is of the opinion that DØ should base its Run II software effort on modern Object Oriented (OO) software techniques. This conclusion was arrived at on the basis of personal experience in small projects, reports from large software projects (generally not HEP) and on the conceptual attractiveness of the OO approach. However, a significant minority of the Working Group felt that a rigorous approach to OO techniques in the Run II software effort would introduce an unacceptable risk of failure in meeting required time lines. This conclusion was based on the lack, to date, of an example of a large HEP experiment which has completed its off-line analysis based on OO techniques, the fact that a major segment of the collaboration still requires training in OO techniques and the fact that there is a significant volume of legacy code from Run 1 of a non-OO nature whose potential for reuse might be degraded by such an OO approach. The two opposing concerns are then, a fear of failure to deliver a working software system for detector turn on in 1999 by committing to a new, not fully proven, technology, versus the fear of being cut-off, both in the near term and for the next decade, from a new, powerful technology by failure to integrate it into the current Run 2 software plan. The following recommendations are a compromise between these two concerns.

Language Recommendation:

- *We recommend that C++ and FORTRAN 90 be the only programming languages used in Run 2 off-line software.*

  *Pure FORTRAN 77 and most of the extensions to it used by DØ in Run 1 are a subset of FORTRAN 90 and therefore are consistent with this recommendation. Those extensions use by DØ which are not consistent, such as $'s in names, the format of Integer*2, Byte, etc. declarations, the non-standard argument passing keywords "%val" and "%ref" and perhaps some others should be converted in an orderly transition to FORTRAN 90. Other FORTRAN functionality not consistent between FORTRAN versions, for example those associated with structures, should not be used except under FORTRAN 90. The constraint on existing C code is that it compile under the C++ compiler. Existing C code not meeting this requirement should be converted.*

- *We recommend that a style guide be developed or adopted for each of the DØ programming languages and that adherence to it be enforced for all code which is to become part of the Run 2 off-line system.*

- *We recommend that only C++ be used for graphical user interfaces, overall program control structures and similar functions, where OO techniques are agreed by all to be highly effective. For the implementation of algorithms (predominantly numerical computation), either C++ or FORTRAN 90 may be used.*

- *We recommend that the mixing of FORTRAN and C++ be done only on a relatively coarse scale. A team responsible for developing a relatively self contained but internally highly connected block of code should usually decide on a single language for that block.*

Having recommended a bilingual approach to off-line computing, an important consideration in determining an appropriate Zebra replacement is that it support as fully and effectively as possible the capabilities of both languages. In particular it should support C++ in a manner which allows as full as possible exploitation of the OO character for which it was selected. The Zebra replacement must also permit access to the persistent event data from both languages and must serve as an effective means by which modules written in one language can access data generated by modules written in the other. The following possibilies were considered in the evaluation of possible Zebra replacements.

1. Retain Zebra as an underlying protocol while adopting and/or developing utility layers to insulate against its undesirable aspects and to give a C++ interface. Such a solution requires recognizing that at some point in the not too distant future, DØ would likely have to assume responsibility for Zebra's maintenance.

2. Identify another existing public domain data handling product used by other collaborations or laboratories which is a good match to DØ's needs. A number of such products exist at various points in their life cycle.

3. Identify a commercial data handling product, perhaps a database or an object oriented database, which would satisfy DØ's needs.

4. Develop or participate in the development of a data handling product specifically tailored to DØ's needs.

Event Data Persistency Mechanism Recommendation:

The following recommendations represent the view of a strong majority of the working group. There was a minority view supportive of a second product which is called out in the last recommendation below as a potential backup to the primary recommendation.

- *We recommend that DSPACK be used as the basis for the persistency mechanism which stores and retrieves DØ event data.*

  DSPACK is a package whose primary author is Ryszard Zybert of the University of Birmingham. It is in active use by NA49, a large CERN experiment and has been adopted for use by the two large detectors at RHIC, STAR and PHENIX. Interfaces to both FORTRAN and C are in general use. A Beta version of a C++ interface is available and has been locally tested. DSPACK makes use of pointers in its API and is capable of storing and retrieving pointers in the persistent data. A mechanism exists by which one can use previously defined DSPACK objects in the definition of new objects. DSPACK, as currently released, does not deal with the data versioning problem but a solution is currently under development. A substantial library of utilities is available, but additional utility development will be required. While DSPACK at one time used Zebra I/O and an XDR based (transportable) I/O layer is being written, it currently does native (non-transportable) I/O. DSPACK currently runs only on UNIX systems, including IBM RS6000 and SGI.

- *We recommend that DØ, or possibly the Fermilab Computing Division in close collaboration with DØ, actively collaborate with the author in the maintenance and evolution of DSPACK.*

  This is the desire of the author of the package and it seems highly desirable from DØ's perspective to have this level of local expertise and ability to control the direction of future evolution.

- *We recommend that an immediate project be undertaken to convert the calorimeter reconstruction code to meet Run 2 standards. This project would include a detailed analysis of the current calorimeter code and its RECO environment, the development of a design for it consistent with the above recommendations and the implementation of that design. An important component of this project would be the near term prototyping of small to modest size segments of the calorimeter code using the new language standards and DSPACK. Such prototyping is expected to consist of the development of modules capable of converting selected Zebra banks to DSPACK objects, the development of a DSPACK implementation of the algorithms creating new objects from those converted objects, and finally the development of modules capable of converting the new DSPACK objects back to Zebra banks. These final banks could then be compared to ones produced by Run 1 calorimeter code. The team responsible for this project should include people concerned with the design of Run 2 RECO, people expert in the existing calorimeter code and people experienced in modern software techniques.*

- *We recommend that an immediate effort be undertaken to design a global DØ Data Model for Run 2, and its implementation using DSPACK. Such a design should specificly take into account ease of programming, the efficiency of code and anticipated access patterns. This effort should interact strongly with above described calorimeter reconstruction code conversion project and should also consult closely with those responsible for handling data in the Level 3 environment.*

- *We recommend that there be a formal review of DSPACK after 4 months of use by DØ. This review should seek to affirm this recommendation based on actual DØ experience. If unacceptable problems have been encountered with DSPACK, this review may advise switching to the backup solution mentioned below. A useful byproduct of this review would be a style guide defining appropriate usage of DSPACK in the DØ offline effort.*

- *We recommend that a small amount of additional study be devoted to ADAMO over the course of the next 4 months so that it can function as a backup solution to which DØ could switch if serious problems are encountered with DSPACK. In the course of our evaluation, we identified ADAMO as a full featured, somewhat more mature product than DSPACK which could also serve as a basis for storing and retrieving event data for Run 2. It was thought by the majority of the committee to be less modern in its function and capability, not being inherently as supportive of an object oriented approach as is DSPACK.*

The above recommendations have been made in the context of differing views of the extent to which DØ should commit to OO techniques. This has resulted in recommendations which deliberately retain substantial (perhaps more than otherwise optimal) flexibility. A specific attraction of DSPACK was that it was felt by many to obtrude little in a C++ context, allowing for the possibility of burying most of its calls in the constructors and destructors of objects. While the virtue of this is somewhat dependent on the extent to which OO methodology is important to DØ and the extent to which C++ is used, it seems generally desirable to keep DSPACK at arms length when possible. This was a goal set for our use of Zebra during Run 1. We were less successful than we would have liked in meeting that goal but hopefully we will be more successful this time with DSPACK. Above all, the Working Group felt that timely and decisive action in implementing the above recommendations was essential in meeting the schedule required by the 1999 beginning of Run 2.