USING A DATA MODEL FROM SOFTWARE DESIGN TO DATA ANALYSIS: WHAT HAVE WE LEARNED?

S. M. FISHER

Rutherford Appleton Laboratory, Didcot, Oxon, U

CERN LIBRARIES, GENEVA

P. PALAZZI

CERN, Geneva, Switzerland

CM-P00059924

The ADAMO data system is being used in a number of particle physics experiments. Experience with it indicates that data modelling is a powerful program design method that extends across the whole software life-cycle, although existing support tools are not yet satisfactory. The entity relationship data of ADAMO can be handled by various programming languages, and can be used effectively in interactive data analysis.

1 The ADAMO data system and its applications

Several hundred scientists are working for the ALEPH particle physics experiment at LEP¹, of which about 60 are developing software that will be used and continue to evolve for some 10–15 years. ADAMO was designed to organize the data from experiments of this scale.

ADAMO is based on the entity-relationship (ER) model of Chen [1]. A data model [2] is a strategy for data organization which includes formally defined data structures, operators to act upon the data and validation procedures to ensure that the data obey the imposed constraints. ADAMO provides all of these features in a form suitable for scientific computing, where numerical algorithms are important.

The ER model is simple, yet powerful. It adopts the view that the world consists of entities (objects that can be distinctly identified and have a fixed number of characteristics called attributes) and relationships, which are associations between entities. The attributes may be restricted to have values within given domains.

It is easy to represent ER data as tables. This provides a natural way to present data in a style which is familiar to the scientist. The table is also the basic structure of a relational database management system to which ADAMO can readily be interfaced.

Full details may be found in the programmers manual [3]. The illustrations (from [4]) shown below in Fig. 1 provide a concise overview:

1. An entity-relationship diagram is drawn to express the fundamental properties of the data. In order to have a complete data description, it is important to consider both the available and required information and how they will be processed. This helps to identify logically distinct subsets of the data, their attributes and relationships.

¹Large Electron Positron collider which is due to start in the Autumn of 1989 at CERN (European Laboratory for Particle Physics), Geneva

Figure 1: Overview of ADAMO: from design to results

2. This diagram is expanded into Data Definition Language (DDL). This contains the same information as the diagram, but includes details which, though important, would clutter the diagram. The DDL is then processed by one of the ADAMO tools to build, validate and store a data dictionary, the compiled form of the DDL which can be readily accessed by the other tools.
3. From the data dictionary, interfaces to different languages for data manipulation can be constructed. One such interface is used with the Table Package (TAP) from FORTRAN. The illustration at the bottom left of Fig. 1 shows the use of the TAP. Routines exist to create and destroy entities and their relationships, to select a subset from an entity set and to move from an entity to related entities. A 'selector' mechanism is used to provide efficient handling of subsets via set operations, along with validation routines to check data consistency.
4. Entities and relationships may be output in tabular form by calling one of the TAP routines, as shown at the bottom right of Fig. 1. Note that the columns 'Rank', 'File' and 'Energy' correspond to attributes in the diagram, the DDL and the code, and that the last column 'Cluster' represents the relationship to the Cluster.

ADAMO was developed within the ALEPH Collaboration where it is now used in several programs, notably in the fields of detector description, data acquisition, event reconstruction and viewing, data analysis, and group administration [5]. More recently, other experiments [6,7,8,9,10] are making use of it as shown in Tab. 1. The last two columns, showing the number of entity sets and relationships, give an indication of size.

Group	Project	#E	#R
ALEPH	ADBS: Detector description	154	160
	GPH: 3-d graphics	25	24
	JULIA: Reconstruction program	284	101
	NADIR: people, computers and networks	33	42
	ONLINE	87	99
DELPHI	Bookkeeping system	25	34
FENICE	All software	64	57
OBELIX	Time of Flight data base	8	7
ZEUS	Offline	120	128
	Online	44	58
ADAMO	Data dictionary	40	50
	Rest of system	37	38

Table 1: Some current projects using ADAMO

2 Data modelling across the software life cycle

The software life cycle identifies the phases of analysis, design, coding and operations through which a software system evolves. At each stage there exist methods to describe the current state of the system and procedures to assist in moving to the next stage. Computer Aided Software Engineering (CASE) tools provide support for the description,

and some offer help in the transition between phases. There must be a clear link back through the chain to the initial analysis. So, in a perfect world, if there is something wrong with the software there must be either an error in the analysis or a damaged link in the chain.

With ADAMO, once the DDL file has been created (manually from the diagram), the entities and relationships may be *used by name throughout the software cycle*. It is for this reason that we see ER as an important design method which extends across the whole life cycle in a consistent and traceable manner. ALEPH made use of the Data Flow Diagram [11] (DFD) during the software analysis stage and of structure charts during design. DFDs model a system as a network of processes with data flowing between them, and structure charts show subroutine-call trees and arguments. The link between the DFD and the structure chart can be formalized, but it is not as clear as the traceability provided by ADAMO.

ALEPH used commercially available CASE tools to draw the various diagrams. None of the tools that ALEPH used were able to make adequate connections between the different types of diagram, nor were they able to show directly the connection with the code. In these circumstances it is not surprising that the DFDs are not, in general, up to date; they cannot be guaranteed to correspond precisely to the existing code. A 'reverse engineering' tool [12], able to generate a structure chart from the code, was written; but the connection with the DFD could not be made.

Most of the tools that we have seen have been inflexible; they were unable to draw ER diagrams of the ADAMO type — only the original Chen ones. This explains why a CASE tool must be able to represent what the user wants, either by providing a set of routines to allow him to build his own diagrams, or preferably by making it rule-based.

The checks made by the tools were found to be inadequate, and interfacing them to some home-made tools proved difficult. Commercially available tools with more open interfaces are clearly needed.

3 Data models and programming languages

Programming languages used for scientific computing, such as FORTRAN, offer neither ER nor tabular data. Conversely, database languages such as SQL have powerful operators to handle tabular data but are not good for numerical work, lacking the variables and the control constructs needed for writing algorithms.

ADAMO makes ER data visible to the programmer in FORTRAN as variables in a common block, produced automatically from the data dictionary; the operations are made by calls to subroutines of the TAP. The actual body of the table containing an entity and its relationships is not directly accessible: it is organized by a hidden memory management package [13,14,15], and only a row of the table at a time sits in the corresponding common block. This approach is similar to that of extending procedural languages to handle tables, such as PASCAL-R [16]. The common block corresponding to the 'Cluster' of Fig. 1 is:

```

INTEGER Cluster,Cluster_9999,Cluster_ID
REAL      Cluster_RC,Cluster_FC,Cluster_Energy,Cluster_DEnergy

COMMON /Cluster/ Cluster,Cluster_ID,
+              Cluster_RC,Cluster_FC,Cluster_Energy,Cluster_DEnergy,
+              Cluster_9999

```

Different ways of accessing the same physical data from FORTRAN or other languages, along with different internal storage formats, are also provided, together with the

conversion tools to go from one format to another. The SQL statement to create a table representing the 'Cluster' entity set is:

```
CREATE TABLE CLUSTER (ID NUMBER(4),
                        RC NUMBER,
                        ENERGY NUMBER,
                        DENENERGY NUMBER);
```

SQL can then be used to manipulate the data.

As an example, the ALEPH detector structure is described with DDL, and out of the corresponding dictionary a relational DB schema is produced. Data are entered in the database using SQL, and subsequently transformed into a Direct Access File, read by the simulation and reconstruction programs. The online system uses the same information, read from the file and converted to VAX-FORTRAN records and structures, to draw the detector.

In database jargon, the way a programmer or user sees the data is called an 'external schema'; an 'internal schema' is the physical storage format, and the logical model of the same data is the 'conceptual schema' [17], which is stored by ADAMO in a dictionary. Tab. 2 summarizes all the language mappings currently offered, along with the corresponding external and internal schemata, showing that the ER model can be used from different programming languages. The various external and internal schemata for the same data are all compatible, as they are generated from the same dictionary.

Language	External Schema		Internal Schema	Remarks
	Data	Operations		
FORTTRAN	FORTTRAN COMMONs and opaque data types	ADAMO TAP	Memory Manager	TAP provides full set of manipulation and validation functions
FORTTRAN	Offsets	Statement functions, BOS and FORTRAN	Memory Manager	Poor protection of data structures and reduced legibility of code
VAX-FORTRAN	Records and structures	VAX-FORTRAN	VAX-FORTRAN	Tables defined at maximum size, non-portable, all operations manual
C	Records and structures	ADAMO TAP	Memory Manager	Relies upon being able to call FORTRAN routines from a C program.
SQL	SQL: CREATE TABLE and INDEX	SQL: INSERT, SELECT etc	RDBMS with SQL.	operations on tables by SQL
EARL	Commands	ADAMO TIP	Memory Manager	

Table 2: ADAMO supported programming languages

4 Interactive data analysis

Consider an ER diagram of which Fig. 2 is a part:

Figure 2: ER Diagram showing Track and Vertex

The boxes are entity sets, and a relationship is represented by the arrow. The double arrow head means that a Vertex is associated with many Tracks, while the bar (|) indicates that it is possible to have a Track with no Vertex.

In order to 'Histogram the momentum of positive tracks with χ^2 per degree of freedom less than 2 and from a vertex within 5 mm of the z axis', if the data were available in a relational DBMS, with SQL as the query language, the following SELECT statement would produce the values of p to be entered into the histogram:

```
SELECT p FROM Track, Vertex
WHERE Track.Charge > 0 AND Track.CHI2/Track.NDF < 2.0
AND ( POWER(Vertex.x,2) + POWER(Vertex.y,2) ) < 25.0
AND Track.Vertex = Vertex.ID;
```

This notation has some drawbacks: all the tables to be used have to be listed at the beginning of the query, and it is not obvious that the next two lines are respectively defining subsets of tracks and of vertices. The last line corresponds to the relationship between Track and Vertex, which in our case is implicit, as it has already been specified in defining the data. After selection, the values of p must be passed to a histogramming package that is not part of current SQL interpreters, so the SELECT statement should be embedded in some procedural language such as FORTRAN.

The EARL language was defined [18] to overcome these limitations, so that the same operation is expressed as:

```
HISTOGRAM Track(p) [Charge > 0 & Chi2/NDF < 2.]
WITH Vertex [x**2 + y**2 < 25.];
```

The use of selectors, identified by name, is encouraged to simplify the queries and because the selectors may then be used repeatedly. The following example defines the subset of 'Good' Tracks, 'GoodPos' Tracks and 'NearZ' Vertices, and then histograms momentum of those 'GoodPos' Tracks related to a 'NearZ' Vertex:

```
SELECT Good Track [CHI2/NDF < 2.];

SELECT GoodPos Good Track [Charge > 0.];

SELECT NearZ Vertex [x**2 + Y**2 < 25.];

HISTOGRAM GoodPos Track(p) WITH NearZ Vertex;
```

This gives the same result as before, but now the selectors may be used again, for example:

HISTOGRAM Vertex(z) WITH >3 GoodPos Track;

A number of EARL interpreters, based on these ideas, have been built [19,10]. Another such interpreter, the TIP (Table Interaction and Plotting), is being developed. It makes use of PAW, the Physics Analysis Workstation package [20], and will become part of ADAMO in the near future. Fig. 3 shows a picture from a workstation running the TIP.

Figure 3: Part of a workstation screen showing the use of TIP

5 Acknowledgements

We are grateful to very many users of ADAMO who by their comments have influenced the development of the system. Alberto Aimar and Giuseppe La Commare have worked enthusiastically to make the TIP a reality. Sarah Wheeler, Andy Belk, Mike Green, John Shade and John Thompson read this paper and made many useful comments and finally Suzy Vascotto helped to make it readable.

References

- [1] P.P. Chen, *The Entity-Relationship Model-Toward a Unified View of Data*, ACM Trans. on Database Systems 1 (1976) p. 9.

- [2] D.C. Tsichritzis and F.H. Lochovsky, *Data Models* (Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982).
- [3] R. Brazioli, S.M. Fisher, P. Palazzi and W.R. Zhao, *The ADAMO programmer's manual*, Aleph Collaboration, CERN.
- [4] M.G. Green, *The ADAMO Data System, an Introduction for Particle Physicists*, Royal Holloway and Bedford New College 89-01 (1989).
- [5] Z. Qian et al., *Use of the ADAMO data management system within ALEPH*, Computer Phys. Commun. 45 (1987) p. 283.
- [6] A. Putzer, *Data Structures in HEP: Modelling and Implementation*, This conference.
- [7] K. Gather, *(SASD) - Tools for Program Design*, This conference.
- [8] H. Kowalski, T. Poser, L. Stanco and E. Tscheslog, *Investigation of ADAMO Performances in the ZEUS Calorimeter Reconstruction Program*, This conference.
- [9] P. Erhard et al., *The ZEUS Data Quality Monitoring System*, This conference.
- [10] F. Etienne and A. Bonissent, *Artificial Intelligence Steering Program for Data Manipulation*, This conference.
- [11] S. M. Fisher, *The Practice of SA-SD*, CERN School of Computing, Oxford, 1988 (CERN '89-06, Geneva, 1989), p. 34.
- [12] J. Bunn, *PRODES, a tool to analyze and describe a FORTRAN program*, ADAMO Application Note 3 (1986).
- [13] R. Brun, F. Carena, H. Grote, M. Hansroul, J-C Lassalle and W. Wojcik, *ZBOOK User's guide and Reference Manual*, CERN Program Library Q210, CERN, DD/US/73.
- [14] V. Blobel, *The BOS system*, DESY Internal Report R1-88-01 (1988).
- [15] R. Brun and J. Zoll, *ZEBRA User Guide*, CERN Program Library Q100.
- [16] J. W. Schmidt, *Some High Level Language Constructs for Data of Type Relation*, ACM Trans. on Database Systems, 2 (1977) p. 247.
- [17] D.C. Tsichritzis and A. Klug, *The ANSI/X3/SPARC DBMS framework report of the study group on database management systems*, Information Systems, 3 (1978).
- [18] S.M. Fisher, *EARL: Entity Relationship Query Language*, ALEPH Note ALEPH 88-45 (1988).
- [19] M. G. Green, *A prototype of the Entity And Relationship Language — EARL*, ADAMO Application Note 13 (1987).
- [20] R. Brun, O. Couet, N. Cremel, C. Vandoni and P. Zancarini, *PAW — Physics Analysis Workstation*, CERN Program Library Q121.