A Conceptual Model of a Nuclear Physics Experiments Database

CLAS-NOTE 92-004

Bryon K. Ehlmann
Dept. of Computer Information Systems, P.O. Box 164, Florida A&M University, Tallahassee, FL 32307, (904) 599-3050, ehlmann@cs.fsu.edu

Lawrence C. Dennis
Dept. of Physics, Florida State University, Tallahassee, FL 32306, (904)644-1804, larry@fsulcd.physics.fsu.edu

Gregory A. Riccardi
Dept. of Computer Science, Florida State University, Tallahassee, FL 32306, (904) 644-2869, riccardi@cs.fsu.edu

February 17, 1992

Abstract

Nuclear physics experiments to be conducted at future accelerators will result in the accumulation of vast quantities of data. This paper briefly discusses an object-oriented database (OODB) approach for effectively managing this data and, more importantly, presents, as an initial by-product of this approach, an object-based, conceptual model of a nuclear physics experiments database. The model is provided by an Object-Relationship Diagram (ORD). The concepts and conventions related to this diagram are explained, and the ORDs for the proposed database are given. The ORD model of a nuclear physics experiments database presented here results from software research and development efforts associated with the Continuous Electron Beam Accelerator Facility (CEBAF).

KEYWORDS: CEBAF Experimental Nuclear Physics Object-oriented databases Object-Relationship Diagram Scientific databases

1 Introduction

The experimental nuclear physics community needs improved software development and data management methodologies and tools. Detectors at the Continuous Electron Beam Accelerator Facility (CEBAF), which is scheduled to be operational in 1994, will collect data at a rate many times that of any current facility, and the data analysis will be significantly more complex. In addition, future projects such as the superconducting supercollider (SSC) and the relativistic heavy ion collider (RHIC) will be orders of magnitude more complex than CEBAF, both in the amount of data and in the data analysis software. Traditional approaches to software development and data management used within the physics community will not suffice to meet the requirements of these projects.

The software research and development efforts currently underway for the CEBAF Large Acceptance Spectrometer (CLAS) provide an immediate incentive and opportunity to begin to experiment with some new approaches. These efforts are being carried out by physicists and professional programmers at CEBAF and by the CLAS Software Collaboration, a small group of university-based physicists and computer scientists. The CEBAF

personnel are responsible for the data acquisition software that collects measurements from the electronics of the CLAS detector. The collaboration is responsible for the detector simulation, data management, and physics analysis software. It is likely that much of the software developed by the collaboration will be used by the other CEBAF detectors.

In developing this software, there are a number of new approaches to be investigated that relate to the physics--e.g., new approaches to track finding. There are also new approaches that relate to computer science and science in general. The computer scientists in the collaboration are especially interested in researching an object-oriented database (OODB) approach to scientific databases. Such an approach combines object-oriented programming concepts with traditional database management system (DBMS) capabilities, which have been used for years in managing large commercial databases [Bert91, Jose91, Khos90, Kim90, Zdon90]. The data management requirements of CLAS offer an excellent opportunity to experiment with such an approach. A CLAS experiments database, like most scientific databases, must facilitate the collection, storage, analysis, update, and sharing of vast quantities of data resulting from experiments conducted over a number of years. The problems of effectively managing such large scientific databases have recently been studied and documented [Fren90, Lagu90, Sdmw90].

An OODB approach to scientific databases in general and CEBAF in particular offers many potential benefits over traditional approaches. Among these benefits are improved software quality, reduced development and maintenance costs, and increased reusability of software by other detectors and future accelerators. Increased speed in data acquisition and analysis may also be achieved through new parallel processing techniques that are based on the OODB model. Finally, the ability of scientists to locate, access, understand, and analyze data may be significantly enhanced. Whether these benefits can be realized for the CLAS database will be determined by prototyping an OODB solution.

One of the first steps in designing this OODB prototype is to develop an object-based, conceptual model of the CLAS experiments database. Object-Relationship Diagram (ORDs) [Ricc91] are used to provide this model. The ORD model identifies and documents all potential object types within the database and the relationships between them. The terminology used to identify object types in the ORD is used in all subsequently developed software and the user interface. The ORD model is conceptual in that objects and relationships are described logically, independently of how they might be physically stored or accessed on the computer. An ORD model is useful even if the CLAS experiments database were developed using a traditional approach.

The major purpose of this paper is to present the ORD model of the CLAS experiments database. Since much of this model is generic to experiments conducted with any particle detector, the model can be viewed as a general model of a nuclear physics experiment database. Standard software development practice demands that a conceptual database model be reviewed by potential users. Their feedback is critical at the modeling stage of database development. Therefore any comments, questions, or suggestions about the model from the experimental nuclear physicist community will be welcomed.

The sections which follow provide additional information on CEBAF and the general requirements for the CLAS experiments database, explain the general concepts and conventions related to the ORD, present the ORD model for the CLAS experiments database, and provide concluding remarks. Some of the technical information given in this paper concerning CEBAF, including diagrams of the CLAS detector, was obtained from CEBAF project documents.

2 CEBAF and the CLAS Experiments Database

CEBAF is currently under construction in Newport News, Virginia. Physicists will perform pure research at CEBAF to develop a better quark-based understanding of

atomic nuclei. The main component of CEBAF is a superconducting electron accelerator with a maximum energy of 4GeV. Its continuous electron beam with a maximum current of 200 microamperes can be simultaneously used for scattering experiments within three experimental halls, or end-stations. One of these, Hall B, will house the CLAS detector.

Of the three CEBAF end stations, the CLAS end station presents the greatest technological challenge in its electronics, physics, and analysis software. A perspective view of this end station is shown in Figure 1. Here the CLAS support structure is not shown and the detector devices have been pulled away to reveal the detector magnet. Figure 2 provides an enlarged view of the CLAS detector magnet and support structure, and Figures 3 and 4 indicate the positioning of detector components within the CLAS detector by giving the longitudinal and cross sections at the target position. The dotted lines in Figure 3 show the projection of the magnet coils. The curved line emanating from the target at position (0, 0) represents the trajectory of a single track. Figure 4 shows how the coils of magnet divide the detector into six sectors.

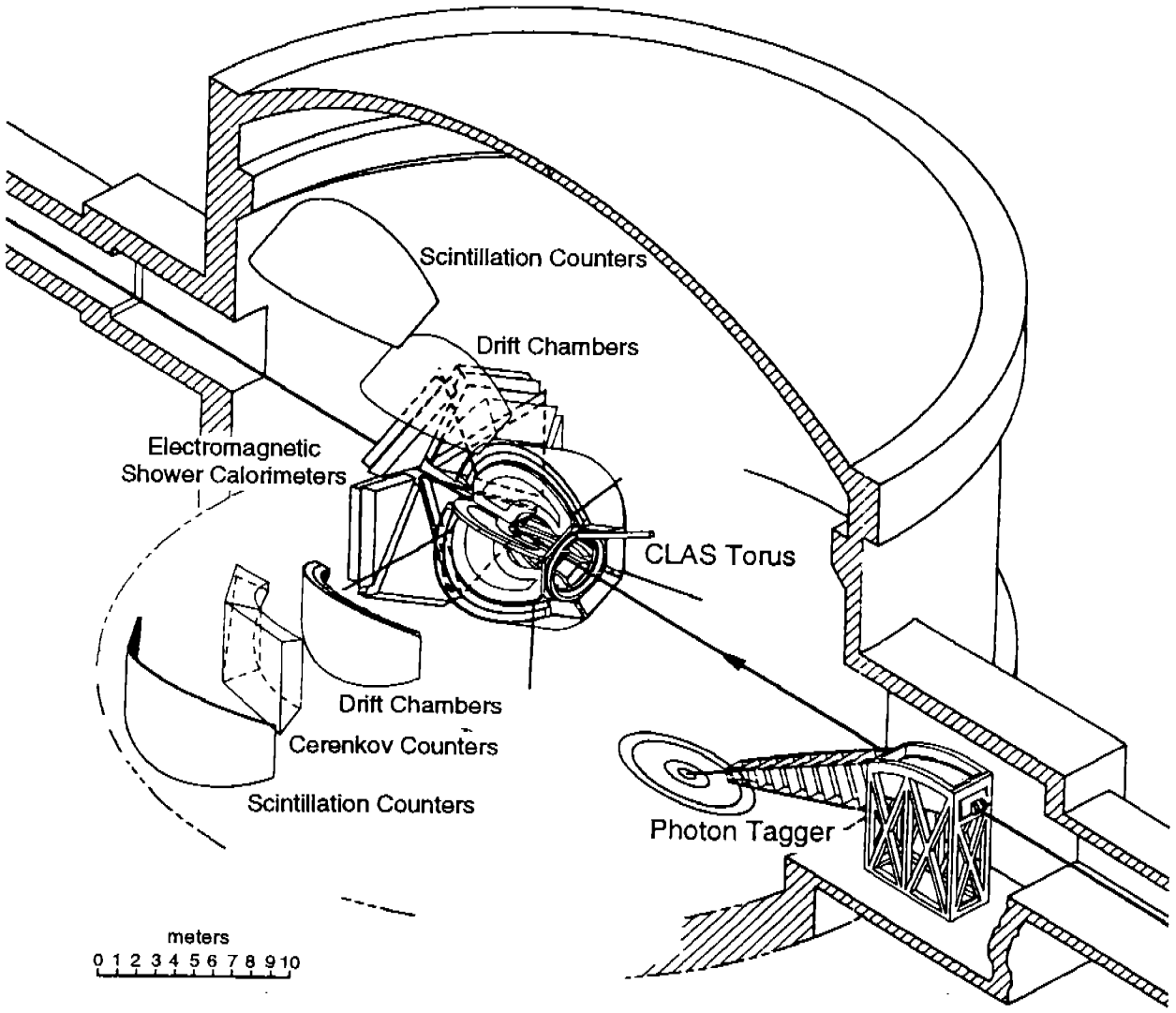

Figure 1. General Arrangement of End Station B and CLAS Detectors

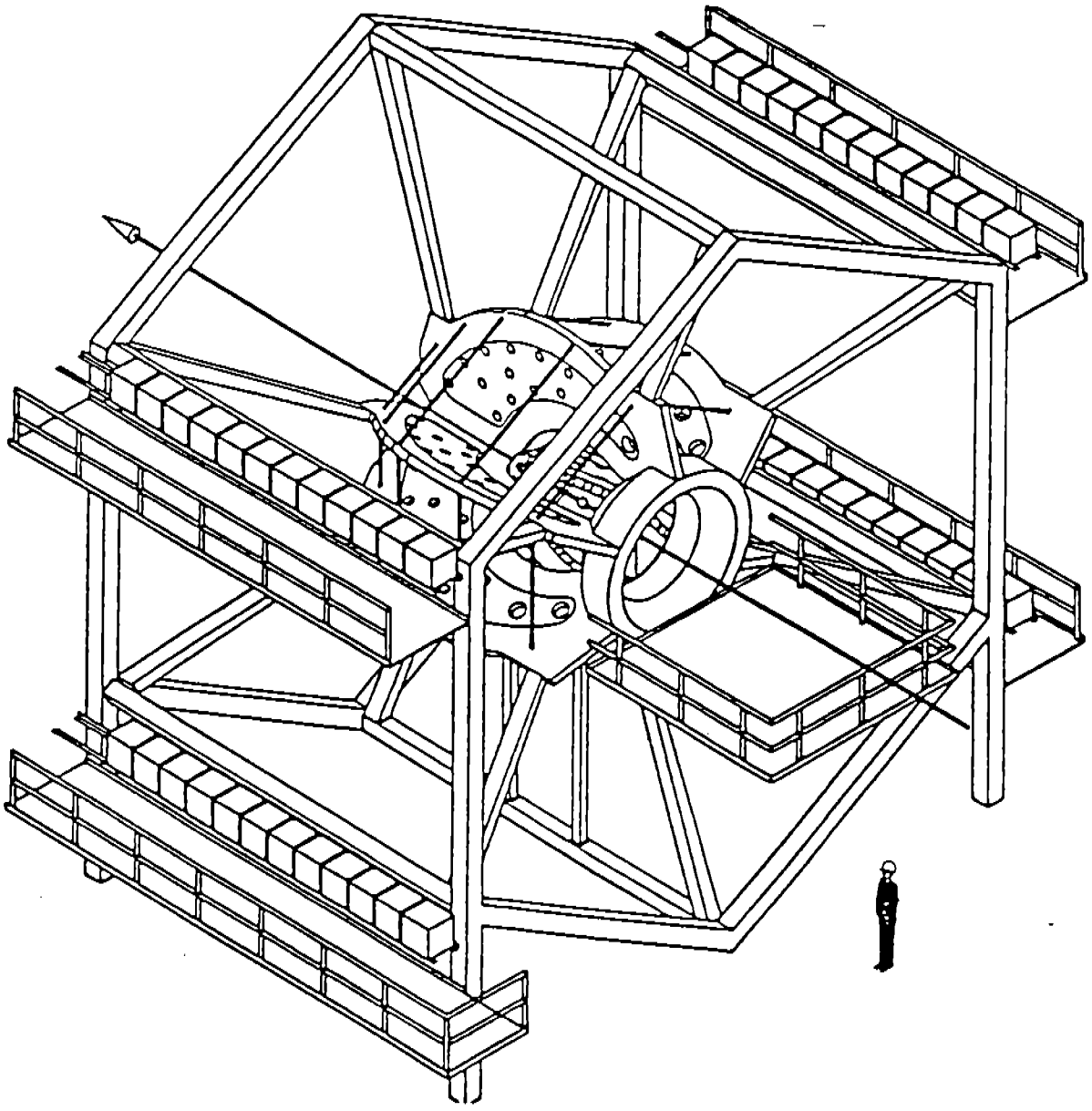

Figure 2. CLAS Detector Magnet and Support Structure

The software required for the CLAS detector is essentially a scientific database application requiring a relatively large database. The CLAS experiments database must manage the data associated with all of the experiments conducted using the CLAS detector. During each "run" of an experiment, data must be collected on the events, or collisions, as they occur. The events, which are made up of numerous "raw hits" on the detector components, will occur at rates in excess of 1000 per second. Roughly, each event will require up to 10K bytes of storage.

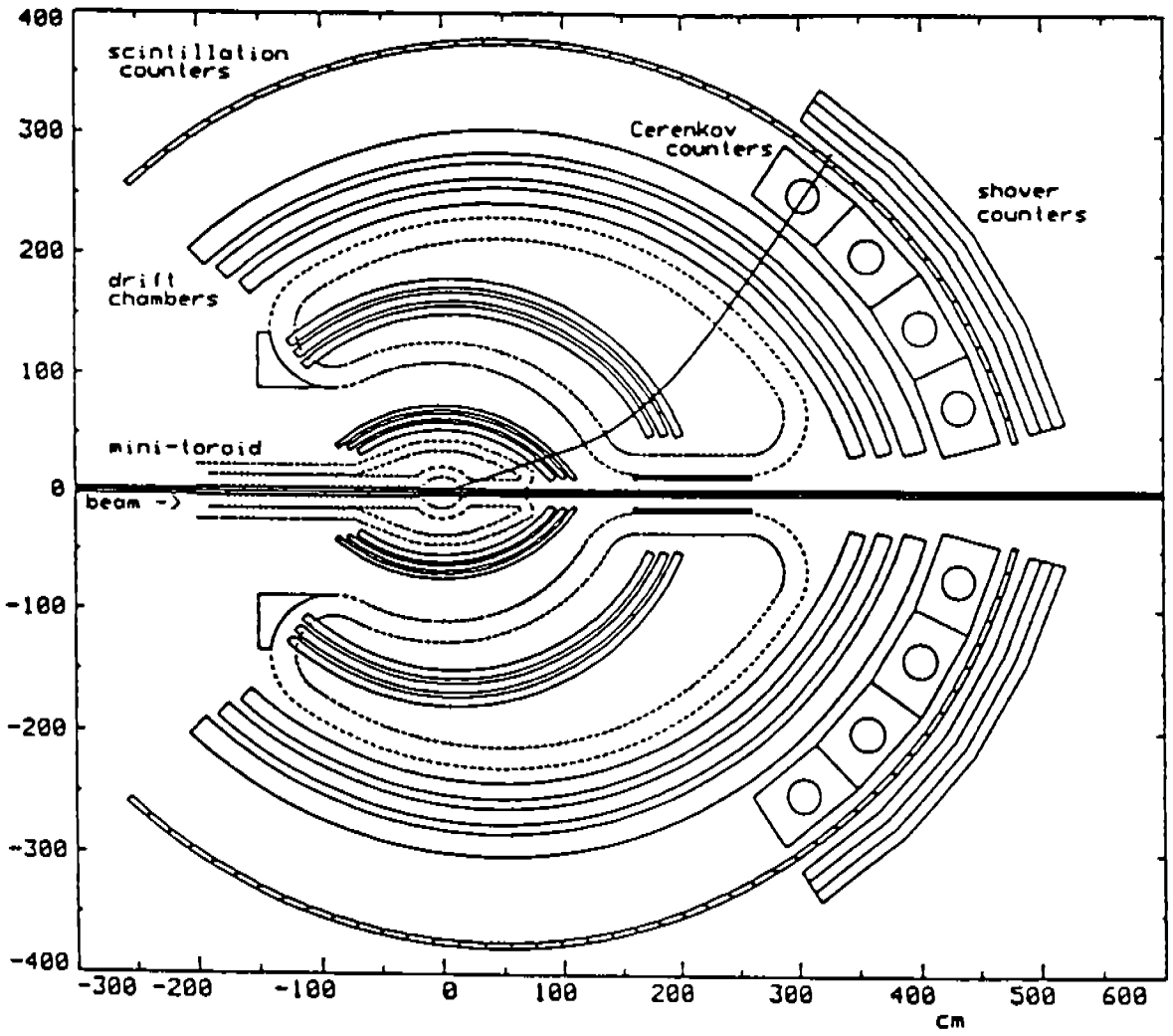

Figure 3. Longitudinal Section of CLAS Detector at Target (Side View)

Data about each run and experiment, the meta-data, must also be recorded. For example, data about the environment and configuration of the detector must be recorded so as to be current at all times during the experiment. This includes the geometry, calibration, and status of all detector components.

Based on the collected data and certain "rules", or programs, provided by the physicists, the raw hits must be filtered and iteratively analyzed to determine track segments, complete tracks, and finally the actual scattered particles associated with each event. During data collection and analysis, the scalar and histogram results desired by the physicist must be computed and recorded for the event, the run, and the experiment. The database must allow physicists other than the original experimenters to access the data at various stages of analysis; to correct environment and configuration data when it seems suspect; to do their own analysis; and to record their assumptions, methods, and results.

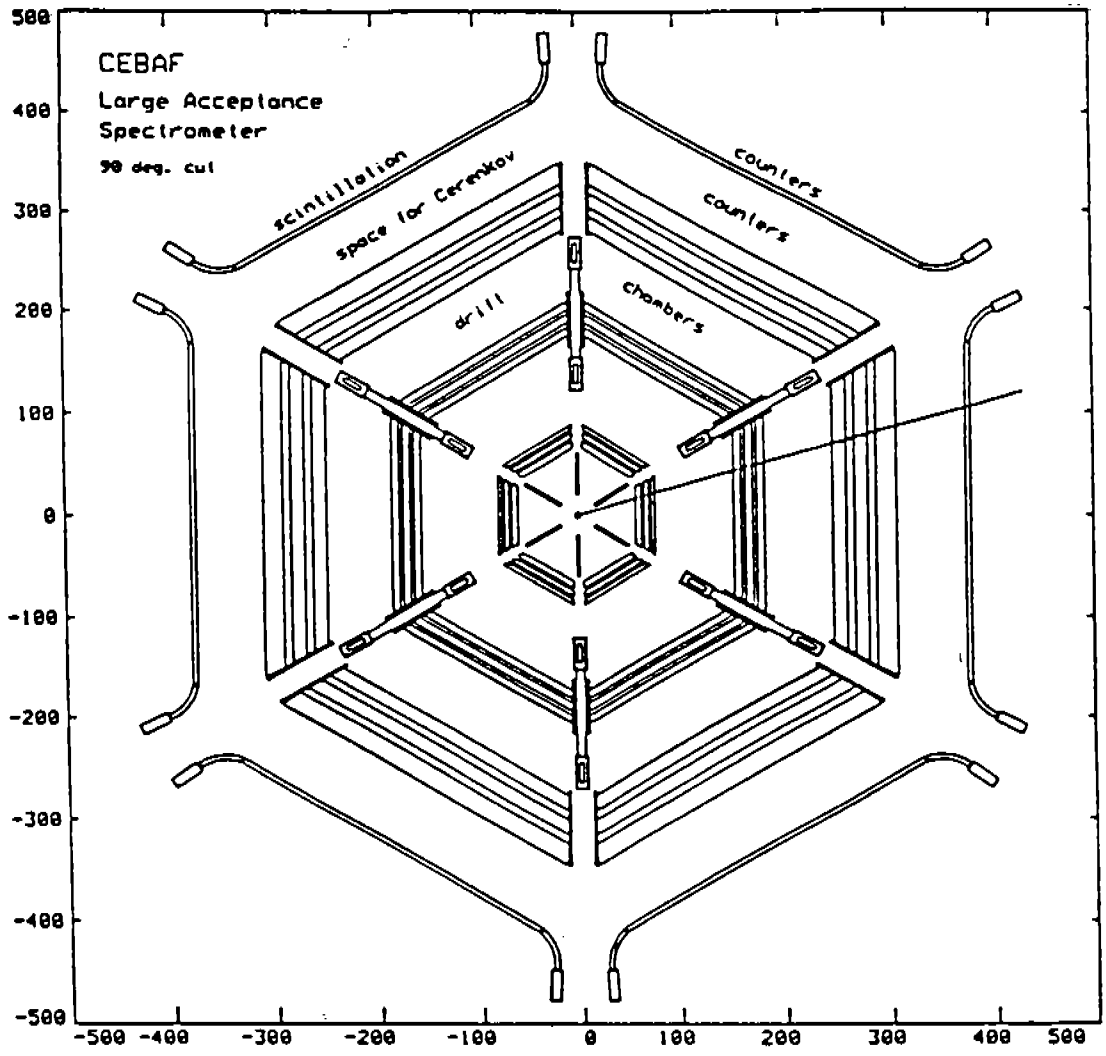

Figure 4. Cross Section of CLAS Detector at Target (Front View)

3 Object-Relationship Diagrams (ORDs)

A valuable first-step in developing a complex database, like the CLAS database, is to construct a conceptual model. An Object-Relationship Diagram (ORD) provides such a model. It is based on the traditional Entity-Relationship Diagram (ERD) [Chen76]. An ORD differs from an ERD in three major ways. First, class attributes are not shown since including them would clutter the diagram. Second, all relationships are binary. And finally, precise cardinalities are given to identify relationship types having specific semantics. Figures 5 through 9 show the ORDs for the CLAS experiments database.

3.1 Concepts and conventions

Rectangles in an ORD identify object classes within the database. An object class defines the set of all objects of a particular type whose common attributes and behavior are of interest in the application. The name given in the rectangle identifies the object class. In Figure 5, EXPERIMENT identifies all objects of type EXPERIMENT. Such objects have common attributes--e.g., a CEBAF assigned id, a beginning date, and a set of runs--and common behavior--e.g., initiation, completion, and display all runs.

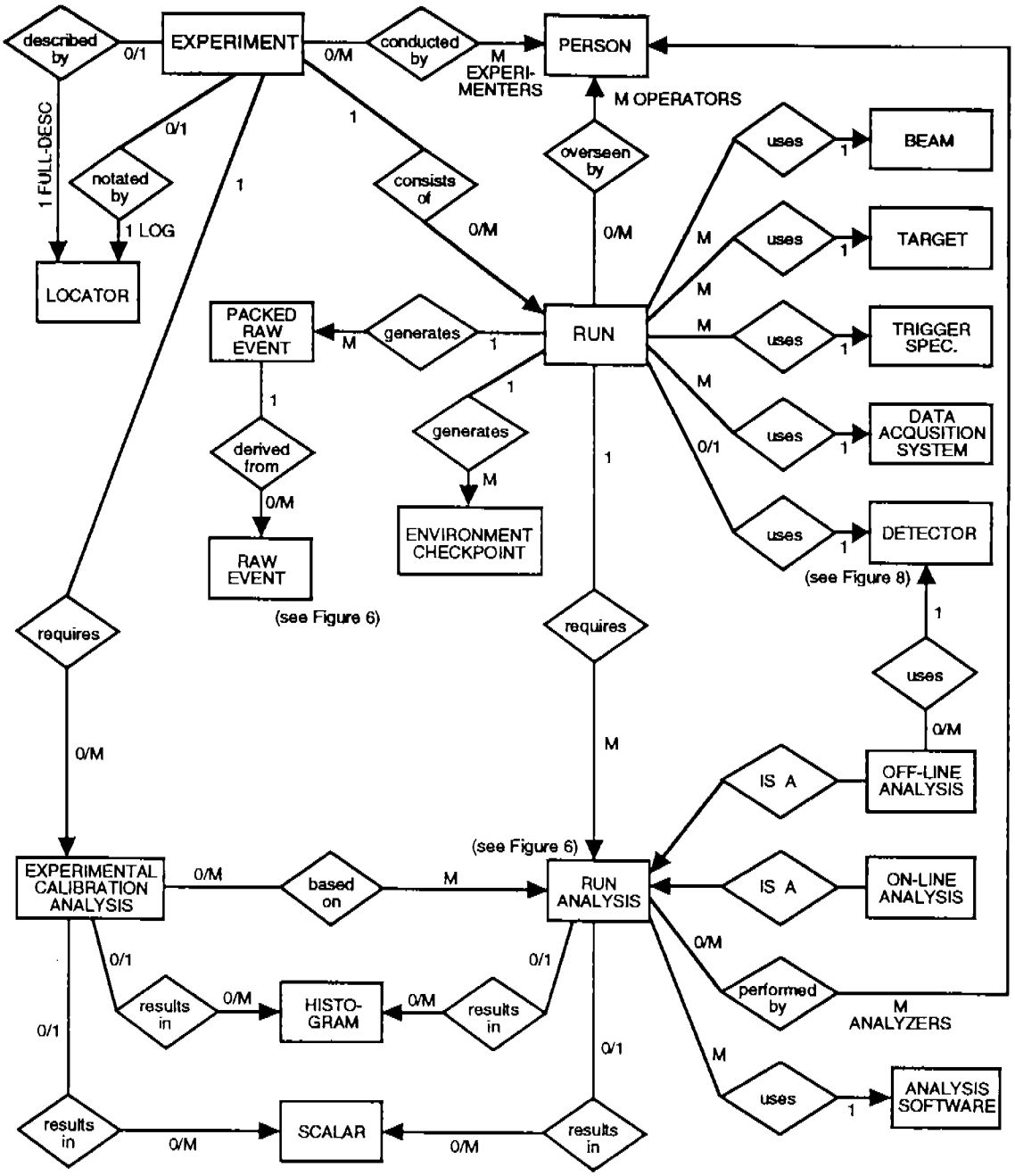

Figure 5. Experiments Overview ORD

Diamonds in an ORD identify relationships. Two rectangles are linked via diamonds to show a relationship between the two object classes. All relationships are binary. Arrows point from the *subject* class of the relationship to the *relative* class, and the relationship name given in the diamond identifies the relationship from this perspective. All relationships are bidirectional. The inverse relationship is obtained by reversing the arrows and supplying a name appropriate to this direction. An inverse relationship name may be given in parenthesis after the relationship name.

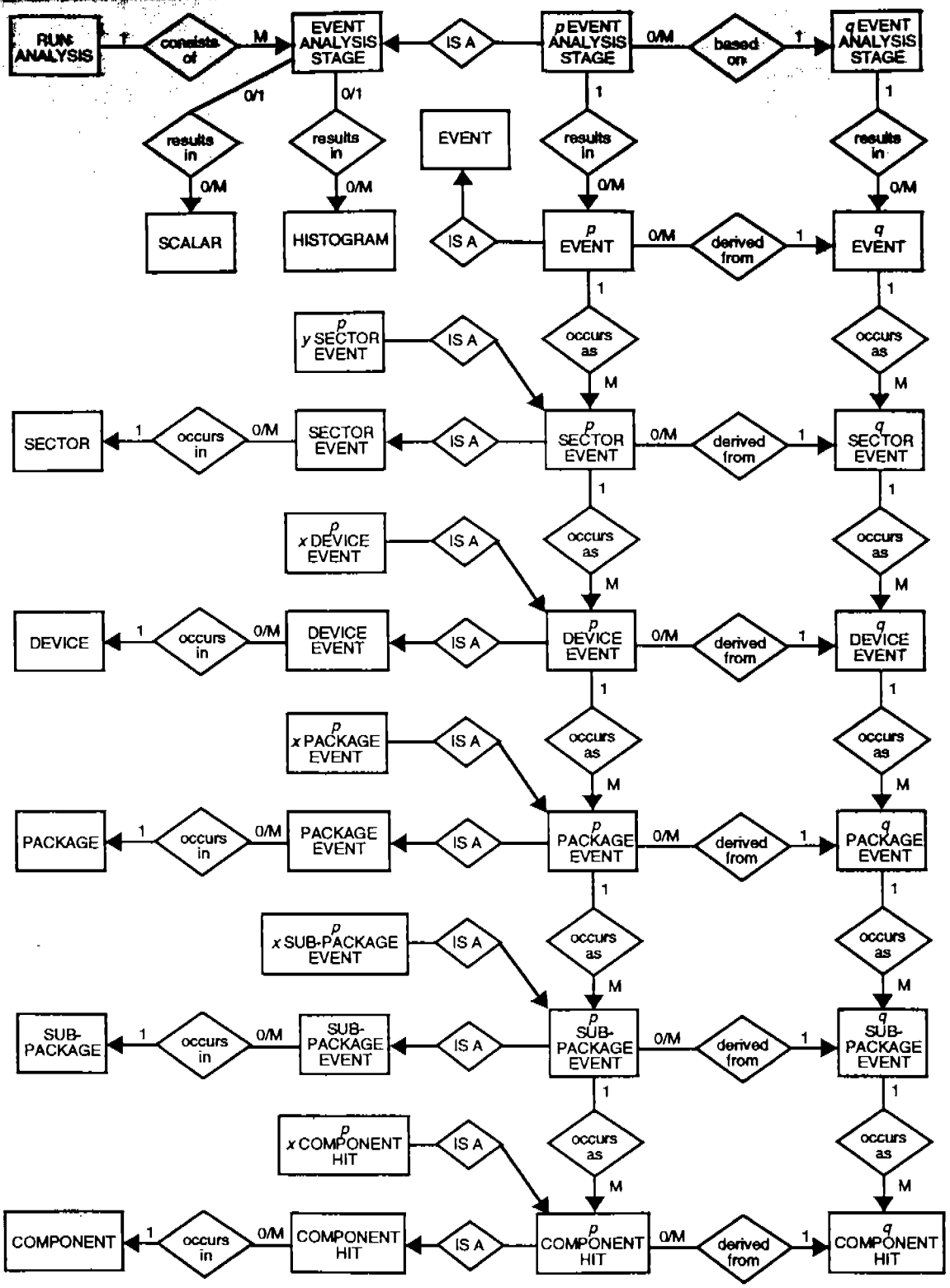

Figure 6. Run Analysis ORD

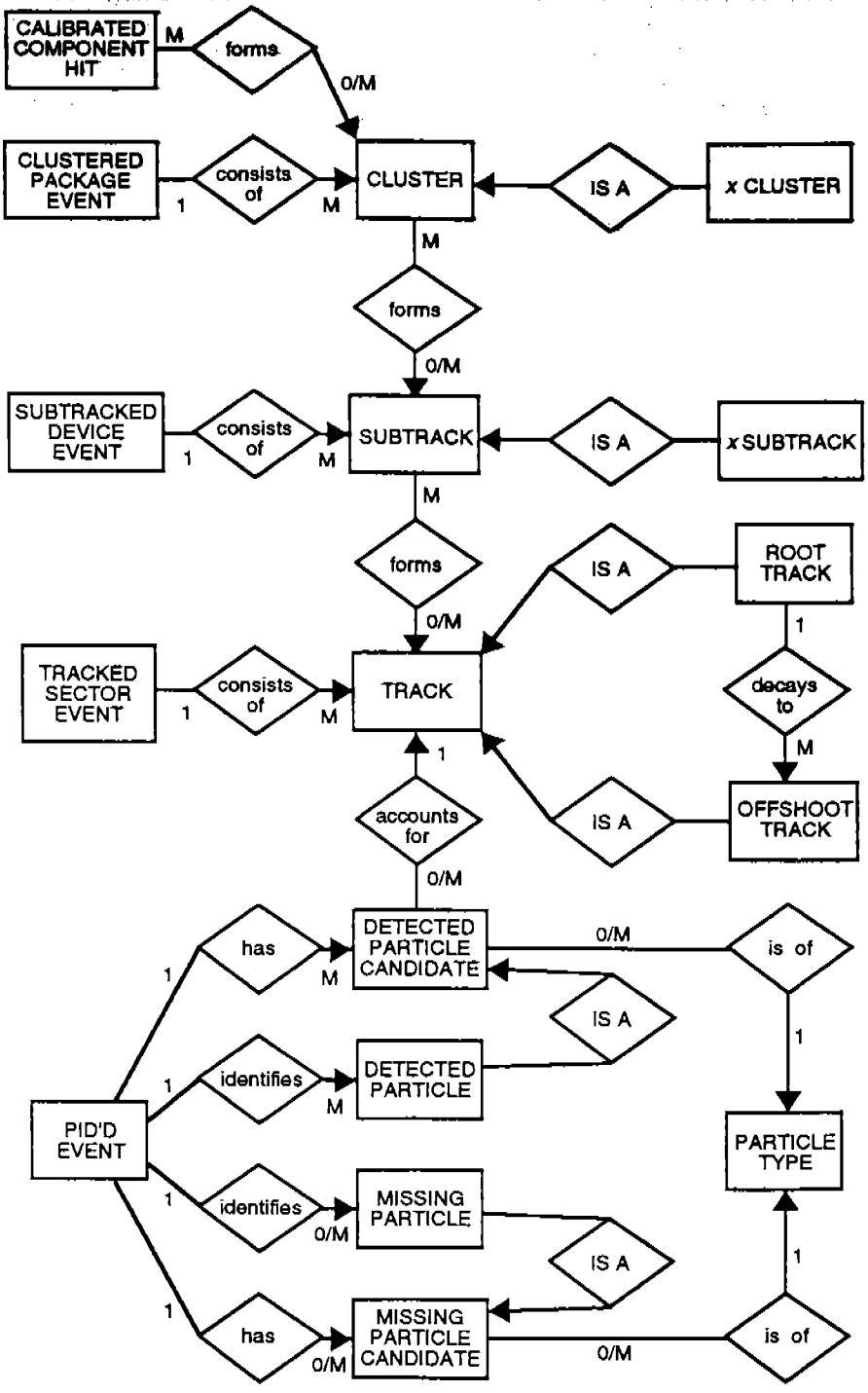

Figure 7. p Event ORD

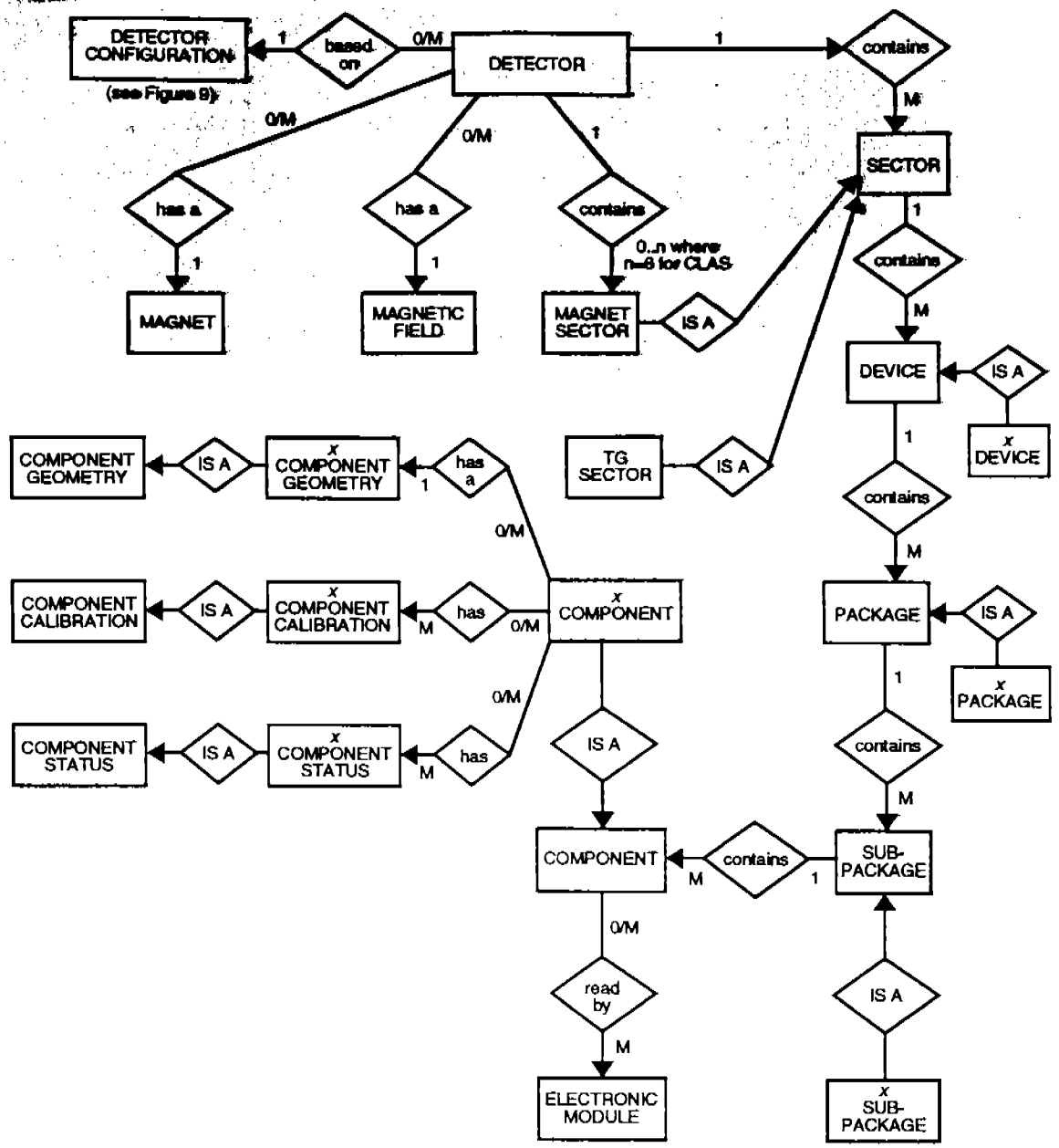

Figure 8. Detector ORD

Cardinalities are given in the ORD to precisely define the type of relationship. The cardinality specification given before the diamond node in the direction of the relationship applies to the subject class; the one given after the diamond applies to the relative class. The relative class cardinality describes the number of objects of the relative class that can relate to a single object of the subject class. Likewise, the subject class cardinality describes the number of objects of the subject class that can relate to a single object of the relative class.

The common cardinality specifications are **0**, **1**, **M**, **0/1**, and **0/M**. The / denotes "or" and the **M** denotes "Many", which is taken to mean one or more.

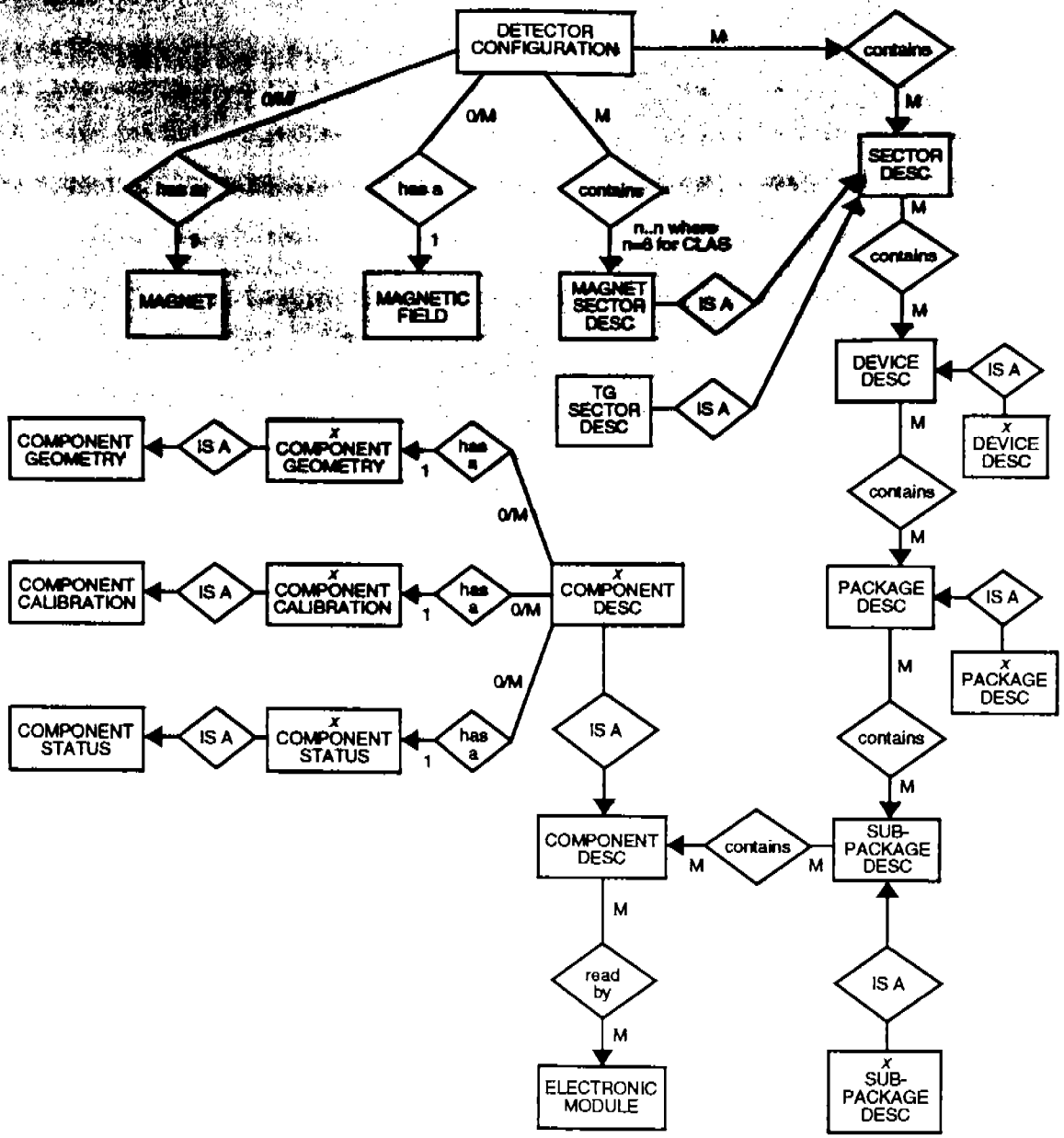

Figure 9 Detector Configuration ORD

Figure 5 shows that an EXPERIMENT "consists of" zero or many RUNs and that a RUN "is part of" a single EXPERIMENT.

A name describing the *role* played by class objects in a relationship may be given after or below the cardinality specification for the class. In Figure 5, an EXPERIMENT is "performed by" one or more PERSONs. The PERSON plays the role of an EXPERIMENTER in this relationship.

An "IS A" is a special type of relationship. It indicates that the subject class is a *subclass* of the related class, the *superclass*. That is, objects of the subject class are also objects of the related class. Objects of a subclass inherit the attributes and behavior associated with its superclass. A subclass can also have its own unique attributes and behavior.

In Figure 5, an OFF-LINE ANALYSIS "is a" kind of RUN ANALYSIS. Cardinality specifications are not given for IS A relationships.

3.2 Relationship types and semantics

The concepts and conventions given in the previous section are sufficient to permit review of the ORDs for the CLAS experiments database; however, a deeper understanding of the ORD model is achieved by understanding the semantics that are associated with the different types of relationships appearing in the diagrams.

Relationship types are defined by the cardinalities assigned to the subject and related classes. For example, the type of the "consists of" relationship in Figure 5 between EXPERIMENT and RUN is defined as 1-to-0/M, and the relationship type of "uses" between RUN and BEAM is M-to-1.

The semantics prescribed for a relationship type place constraints on the state of a database and specify that certain database operations perform checks and have desirable side effects in order to maintain these constraints. Affected operations are:

1) add a class object,
2) delete a class object,
3) create a relationship, i.e., relationship instance, between a two class objects,
4) destroy a relationship between two class objects, and
5) change a relationship for a class object so that it relates to different class object.

These operations operate at a higher level than the object or record level operations typical of most database systems. For example, the delete of an object can cause the automatic deletion of many related objects.

The semantics of a relationship type are derived from the semantics of the *cardinality* for the subject class and the relative class. For example, the semantics of a 1-to-0/M relationship type are the semantics of a -to-0/M specification plus the semantics of a 1-to- specification. *Cardinality* semantics are given below in terms of a relationship R having the given *cardinality* for the relative class. The semantics in terms of an inverse relationship having the *cardinality* for the subject class are obtained by substituting the terms "relative" for "subject" and "subject" for "relative" wherever they occur below.

-to-1 (inverse 1-to-). The generic name for relationship R in the direction of the relative class is "defined by a." Each subject class object depends on a relative class object for its definition and existence. A subject class object cannot be added without having an R relationship with a relative class object. A subject object must be deleted if its relative object is deleted or the its R relationship is destroyed (either directly or by a relationship change for the relative object).

-to-M (inverse M-to-). The generic name for relationship R in the direction of the relative class is "defined by one or more." Each subject class object depends on one or more relative class objects for its definition and existence. A subject class object cannot be added without having an R relationship with at least one relative class object. A subject object must be deleted if its only relative object is deleted or its only R relationship is destroyed.

-to-0/1 (inverse 0/1-to-). The generic name for relationship R in the direction of the relative class is "may be described by a." A subject class object does not depend on a relative class object for its definition and existence. Instead, a relative class object serves only to further describe the subject class object. A subject object is never implicitly deleted as a result of the *cardinality* of its relative class. If a relative class object is deleted, any R relationships that it has with subject class objects must be destroyed.

-to-0/M (inverse 0/M-to-). The generic name for relationship R in the direction of the relative class is "may be described by one or more." A subject class object does not depend on a relative class object for its definition and existence. Instead, relative class objects serve only to further describe the subject class object. A subject object is never implicitly deleted as a result of the *cardinality* of its relative class. If a relative class object is deleted, any R relationships that it has with subject class objects must be destroyed.

The four possible *cardinality* descriptions for the two classes in a relationship result in ten distinct and basic relationship types. (Six of the sixteen possible combinations are nondistinct since relationships are bidirectional.) Examples of many of these types appear in the given ORDs. Relationship types are more finely tuned when additional specifications are given that define the level of binding, i.e., the destructibility of relationship instances and the implicit deletability of objects based on the relationship. Such specifications are the next step in the design process and are beyond the scope of this paper.

The semantics given above can be applied in reviewing the ORDs in Figures 5 through 9. For example, Figure 8 indicates that a SUBPACKAGE "contains" many COMPONENTs and that this relationship is 1-to-M. The semantics of the 1-to-M relationship type dictate that if a SUBPACKAGE is deleted, all related COMPONENTs must be deleted. The semantics of this relationship also dictate that if the last COMPONENT is removed from a SUBPACKAGE , i.e., its relationship with a SUBPACKAGE is destroyed, the SUBPACKAGE must be deleted. Again, whether a SUBPACKAGE can actually be deleted or the last COMPONENT in a SUBPACKAGE can be removed, is determined by additional specifications.

The relationship "consists of" in Figure 5 between EXPERIMENT and RUN is 1-to-0/M. This indicates that an EXPERIMENT may exist without any RUNs; e.g., the experiment has been setup, but the first run has not yet been initiated. A RUN, however, cannot exist without an EXPERIMENT. Deleting all RUNs for an EXPERIMENT by destroying their relationships with the EXPERIMENT will not delete the EXPERIMENT. If, however, an EXPERIMENT is deleted, all RUNs for that EXPERIMENT must also be deleted.

Finally, a RUN "uses" a type of electron or photon BEAM and the relationship is M-to-1. If all RUNs that use a particular BEAM type are deleted, the BEAM type must be deleted. Deleting a BEAM type, must result in the deletion of all RUNs that use the BEAM type. The explicit deletion of a BEAM type is therefore a good example of a deletion that subsequent specifications would disallow.

4 ORD Model for CLAS Experiments Database

The ORD model for the CLAS experiments database is given by Figures 5 through 9, which are effectively linked together by the object classes that appear in more than one figure. Table 1 lists alphabetically the names of every object class appearing in Figures 5 through 8 and provides a short description for each class. The object classes in Figure 9 are similar to those in Figure 8 are explained in the discussion below. Additional insight into the meaning of the object classes that model the CLAS detector may be gleaned from again reviewing Figures 1 through 4.

Figure 5 provides an overview of experiments. It shows the basic relationships between experiments, runs, packed raw events, experiment calibration analyses, run analyses, and detectors.

Figures 6 and 7 show the objects and relationships created as a result of a run analysis. In each stage of a run analysis new knowledge is gained about each event. This knowledge results in either the elimination of the event from further analysis or a new type of event that references the previous type and records the new knowledge gained.

Table 1. Object Class Descriptions

Object Class	Description
ANALYSIS SOFTWARE	Description of the analysis programs and routines
BEAM	Description of a beam in terms of type and energy
CLUSTER	A segment of a track detected within a detector package
COMPONENT	Electronic device for detecting a moving particle at a particular location within the detector
COMPONENT CALIBRATION	Parameters describing the calibration of a component at a point in time
COMPONENT GEOMETRY	Parameters describing the spatial orientation of a component at a point in time
COMPONENT HIT	An event as viewed in a particular component, i.e., the detection of a moving particle by a component
COMPONENT STATUS	Parameters describing the status of a component at a point in time
DATA ACQUISITION SYSTEM	Description of the data acquisition system
DERIVED SCALAR	Numeric attribute describing collision
DETECTED PARTICLE	Identified scattered particle
DETECTED PARTICLE CANDIDATE	Possible scattered particle
DETECTOR	Dynamic description of the detector assumed during a run analysis
DETECTOR CONFIGURATION	Static description of the fully configured detector
DEVICE	A collection of similar types of components, e.g., drift chambers, within a sector arranged to detect and identify scattered particles
DEVICE EVENT	An event as viewed in a particular detector device
ELECTRONIC MODULE	Electronic device containing readouts for a number of components
ENVIRONMENT CHECKPOINT	Physical measurements taken at a point in time during a run
EXPERIMENT	An experiment conducted on the detector facility having a facility assigned Id
EXPERIMENT CALIBRATION ANALYSIS	An analysis of runs to determine proper device calibrations
EVENT	Detected collision indentified by an Event Id
EVENT ANALYSIS STAGE	A stage in a run analysis that involves event selection and/or event reconstruction and produces results that can be further analyzed
HISTOGRAM	Result consisting of multiple variables and their values represented as n-tuples
LOCATOR	Identifier for locating a file within a network
MAGNET	Description of the detector's magnet
MAGNETIC FIELD	Description of the field produced by the detector's magnet
MAGNET SECTOR	A geometrical portion of the detector's magnetic field where particles can be detected
MISSING PARTICLE	Particle resulting from collision that was undetected, i.e., no track was found
MISSING PARTICLE CANDIDATE	Possible particle resulting from collision that was undetected
OFF-LINE ANALYSIS	Analysis of the results of a run that is performed after the run has taken place
ON-LINE ANALYSIS	Analysis of the results of a run that is performed during the run, i.e., in real time
p COMPONENT HIT	A *p* event as viewed in a particular component, i.e., the detection of a moving particle by a component
p DEVICE EVENT	A *p* event as viewed in a particular detector device
p EVENT	An event represented by analysis results that are indicated by the term *p*, e.g., a CALIBRATED EVENT is an event that has been calibrated
p EVENT ANALYSIS STAGE	An event analysis stage of type *p* that results in *p* events
p PACKAGE EVENT	A p event as viewed in a particular detector package

Table 1. Object Class Descriptions--continued

Object Class	Description
p SECTOR EVENT	A *p* event as viewed in a particular sector
p SUBPACKAGE EVENT	A *p* event as viewed in a particular detector subpackage
p x COMPONENT HIT	A *p* event as viewed in a particular component of type *x*
p x DEVICE EVENT	A *p* event as viewed in a particular detector device of type *x*
p x PACKAGE EVENT	A *p* event as viewed in a particular detector package of type *x*
p y SECTOR EVENT	A *p* event as viewed in a particular sector of type *y*
p x SUBPACKAGE EVENT	A *p* event as viewed in a particular detector subpackage of type *x*
PACKAGE	A portion of a device, e.g., a drift chamber superlayer, used to detect particle track segments within a sector
PACKAGE EVENT	An event as viewed in a particular detector package
PACKED RAW EVENT	A detected collision represented by the digitized data readout from electronic modules
PARTICLE TYPE	A particular type of particle, e.g., an electron
PERSON	An individual associated with an experiment, its design, actual operation, or analysis
RUN	A continuous operation of the detector over a relatively short period of time using a particular beam and target
RUN ANALYSIS	An analysis of the data collected during a run or of the intermediate results of another RUN ANALYSIS
SCALAR	Result consisting of a single variable and its value
SECTOR	A physical portion of the detector
SECTOR EVENT	An event as viewed in a particular sector
SUBPACKAGE	A portion of a package, e.g., a drift chamber layer, which represents a physical arrangement of components.
SUBPACKAGE EVENT	An event as viewed in a particular detector subpackage
SUBTRACK	A portion of a track detected within a single detector device
TARGET	Description of the material containing the targeted nuclei
TG SECTOR	Sector representing the photon tagger
TRACK	A complete track detected within a detector sector
TRIGGER SPEC.	Specifications that state the conditions under which electronic data will be readout as a raw event
x CALIBRATED HIT	Calibrated hit on a component of a particular type.
x CLUSTER	Cluster of a particular type.
x COMPONENT	Component of a particular type, e.g., a drift chamber wire or a cerenkov counter
x COMPONENT CALIBRATION	Parameters describing the calibration of a component of a particular type at a point in time
x COMPONENT GEOMETRY	Parameters describing the spatial orientation of a component of a particular type at a point in time
x COMPONENT STATUS	Parameters describing the status of a component of a particular type at a point in time
x DEVICE	Device of a particular type, e.g., all drift chamber superlayers within a sector
x PACKAGE	Package of a particular type, e.g., a drift chamber superlayer within a sector
x RAW HIT	Raw hit on a component of a particular type
x SUBPACKAGE	Subpackage of a particular type, e.g., a drift chamber layer within a sector
x SUBTRACK	Subtrack within a particular type of device
y SECTOR	A sector of a particular type, e.g., magnet

Object class names that begin with one or more variables, e.g., *p* EVENT ANALYSIS STAGE in Figure 6, indicate that a number of different object classes exist which have "IS A" relationships with a given superclass, e.g., EVENT ANALYSIS STAGE. For the CLAS experiments database, $p \in E$ where E is the ordered 9-tuple (PACKED RAW, RAW, CALIBRATED, CLUSTERED, SUBTRACKED, TRACKED, PID'D, SELECTED, FINAL), which identifies the different stages of event analysis. The subclasses obtained by substituting appropriate values for the variable, e.g., CALIBRATED for *p*, have attributes, relationships, and behavior specific to their type, e.g., a CALIBRATED EVENT ANALYSIS STAGE results in CALIBRATED EVENTS. In addition, these subclasses inherit the attributes, relationships, and behavior of their superclasses. For example, a CALIBRATED EVENT ANALYSIS STAGE, like any EVENT ANALYSIS STAGE, is part of a particular RUN ANALYSIS and may result in certain SCALARs and HISTOGRAMs.

Figure 6 is the most complicated of the ORDs because of its extensive use of variables in object class names. The following explanations are needed to supplement and clarify this ORD.

1. The EVENT ANALYSIS STAGEs that a RUN ANALYSIS "consists of" are a p_1 EVENT ANALYSIS STAGE, p_2 EVENT ANALYSIS STAGE, ..., p_M EVENT ANALYSIS STAGE, where for CLAS $p_i \in E$, as previously defined, $1 \le i \le M$, $1 \le M \le 9$, and p_i=successor(p_{i-1}) for $2 \le i \le M$. For the first RUN ANALYSIS of a RUN, which is always an ON-LINE ANALYSIS (see Figure 5), p_1 = PACKED RAW.

2. Except for the PACKED RAW EVENT ANALYSIS STAGE of an ON-LINE ANALYSIS, a *p* EVENT ANALYSIS STAGE is "based on" the results of a *q* EVENT ANALYSIS STAGE, where $p \in E$, $q \in E$, $p = q$ or p = successor(q), and the *p* EVENT ANALYSIS STAGE and *q* EVENT ANALYSIS STAGE are related (via a RUN ANALYSIS) to the same RUN. In the context of a single RUN ANALYSIS and using the notation of 1. above, a p_i EVENT ANALYSIS STAGE is "based on" a p_{i-1} EVENT ANALYSIS STAGE, $2 \le i \le M$. If the RUN ANALYSIS is an ON-LINE ANALYSIS, p_1 = PACKED RAW and the p_1 EVENT ANALYSIS STAGE is "based on" the unanalyzed data of the RUN. If the RUN ANALYSIS is an OFF-LINE ANALYSIS, the p_1 EVENT ANALYSIS STAGE is based on the results of an EVENT ANALYSIS STAGE for some other RUN ANALYSIS.

3. A *p* EVENT of a *p* EVENT ANALYSIS STAGE is "derived from" a *q* EVENT of a *q* EVENT ANALYSIS STAGE, where *p* and *q* are as defined above. A PACKED RAW EVENT of a PACKED RAW EVENT ANALYSIS STAGE for an ON-LINE ANALYSIS is "derived from" an unanalyzed PACKED RAW EVENT generated by the RUN (see Figure 5). The *p* EVENT and related *q* EVENT must have the same EVENT Id and relate to the same RUN. Derivation of an EVENT ranges from simple selection to complex analysis.

4. A *p* EVENT "occurs as" one or more *p* SECTOR EVENTs whenever $p \in$ (RAW, CALIBRATED, CLUSTERED, SUBTRACKED, TRACKED). A *p* SECTOR EVENT is "derived from" a *q* SECTOR EVENT whenever $p \in$ (CALIBRATED, CLUSTERED, SUBTRACKED, TRACKED) or $p = q$ = RAW.

5. *y* is a SECTOR type. For CLAS, $y \in$ (MAGNET, TG). TG refers to the Photon Tagger. The database can be extended to include other types of sectors.

6. A *p* SECTOR EVENT "occurs as" one or more *p* DEVICE EVENTs whenever *p*

$\in$ (RAW, CALIBRATED, CLUSTERED, SUBTRACKED). A p DEVICE EVENT is "derived from" a q DEVICE EVENT whenever $p \in$ (CALIBRATED, CLUSTERED, SUBTRACKED) or $p = q =$ RAW.

7. x is a DEVICE, or detection device, type. For CLAS, $x \in$ (EC, SC, CC, DC, TG). EC refers to Electronic Calorimeter, SC to Scintillator Counter, CC to Cerenkov Counter, DC to Drift Chamber, and TG to Photon Tagger. The database can be extended to include other types of detection devices.

8. A p DEVICE EVENT "occurs as" one or more p PACKAGE EVENTs whenever $p \in$ (RAW, CALIBRATED, CLUSTERED). A p PACKAGE EVENT is "derived from" a q PACKAGE EVENT whenever $p \in$ (CALIBRATED, CLUSTERED) or $p = q =$ RAW.

9. A p PACKAGE EVENT "occurs as" one or more p SUBPACKAGE EVENTs and a p SUBPACKAGE EVENT "occurs as" one or more p COMPONENT HITs whenever $p \in$ (RAW, CALIBRATED). A p SUBPACKAGE EVENT is "derived from" a q SUBPACKAGE EVENT and a p COMPONENT HIT is "derived from" a q COMPONENT HIT whenever $p =$ CALIBRATED or $p = q =$ RAW.

Figure 7 provides additional details on the objects that comprise PID'D EVENTs, TRACKED SECTOR EVENTs, SUBTRACKED DEVICE EVENTs, and CLUSTERED PACKAGE EVENTs and the relationships between these objects. These objects are created as event analysis proceeds from the CALIBRATED to the PID'D stage.

Figure 8 shows the objects and relationships that represent the data about the detector used by a run or assumed by a run analysis. Components are organized by subpackage, package, device, and sector. A detector may represent a subset of the completely configured detector, e.g., only certain magnetic sectors or devices are included. A detector is initialized based on a detector configuration, which is discussed below, or another detector. Status, geometry, and calibration data for a detector may be changed by a run analysis. Status and calibration data can even change over the course of a run.

Figure 8 shows relationships involving component geometry, calibration, and status objects and x COMPONENT objects. Analogous objects and relationships exist for x SUBPACKAGE, x PACKAGE, x DEVICE, and y SECTOR objects where x and y are as defined previously for Figure 6. For example, an x PACKAGE "has a" x PACKAGE GEOMETRY that "IS A" PACKAGE GEOMETRY. These objects and relationships are not shown in Figure 8 in order to simplify the ORD.

Figure 9 shows the objects and relationships that represent detector configurations. A detector configuration provides an up-to-date description of the fully configured detector that is independent of any experiment and is used to initialize a detector for an experiment run. When required, changes to the configuration, i.e., changes in structure, status, geometry, or calibration, are made by authorized facilities personnel, not by experimenters, and result in a new detector configuration. Objects within the configuration that are unchanged are referenced by both the old and new configurations. Detector configurations are archived and never deleted.

The object classes SECTOR DESC, DEVICE DESC, PACKAGE DESC, SUBPACKAGE DESC, and PACKAGE DESC and their subclasses shown in Figure 9 are not given in Table 1. They are similar to SECTOR, DEVICE, PACKAGE, SUBPACKAGE, and PACKAGE and their subclasses, respectively; however, there are significant differences that are difficult to convey in a short description. First, each DESC type object, rather than used to describe a single detector, may be used to describe multiple detector configurations. For example, a COMPONENT DESC, rather than being part of a single package, may be part of many package descriptions (for the same physical package) which are used to describe different detector configurations. Second, each DESC type

object, contains a time and date to indicate when the object was created, i.e., the effective date of the description. And finally, each DESC type object, contains only a single status and calibration. When the calibration, for example, changes for a COMPONENT object, a new COMPONENT CALIBRATION object is simply created that relates to the COMPONENT object. When, however, the calibration changes for a COMPONENT DESC object, a new COMPONENT CALIBRATION object is created that relates to a new COMPONENT DESC object. The new COMPONENT DESC object is made part of a new SUBPACKAGE DESC object, which is made part of a new PACKAGE DESC object, which is made part of a new DEVICE DESC object, which is made part of a new SECTOR DESC object, which is made part of a new DETECTOR CONFIGURATION object.

Similar to Figure 8, Figure 9 shows relationships involving component geometry, calibration, and status objects and *x* COMPONENT DESC objects. Again, analogous objects and relationships exist for *x* SUBPACKAGE DESC, *x* PACKAGE DESC, *x* DEVICE DESC, and *y* SECTOR DESC objects. For example, an *x* PACKAGE DESC "has a" *x* PACKAGE GEOMETRY that "IS A" PACKAGE GEOMETRY. These objects and relationships are not shown in Figure 9 in order to simplify the ORD.

In reviewing the ORD model shown in Figures 4 through 9 the reader should keep in mind that it is a conceptual, or logical, model of the database. Decisions such as physical storage media or how, or whether, the database will be physically divided into multiple databases are not reflected by the model.

5 Concluding Remarks

The model for the CLAS experiments database presented in this paper is currently being used to develop two prototypes. A traditional implementation approach is being employed to develop a prototype in Fortran. This prototype assumes that the data and the configuration information required for data analysis are stored in a number of files, with application-specific programs to read, write, and manipulate them. The purpose of this prototype is to validate the ORD data model and to provide an initial framework for the development of the data analysis software.

The second prototype will use a commercially available OODB system that is based on the C++ programming language. The OODB system will be extended as needed with the capabilities required to support the CLAS experiments database. The OODB prototype will integrate the data and the data analysis software, written in Fortran, into the scientific database. Database and query languages will control the various data acquisition and analysis operations, maintain the metadata and all of the raw and derived data, and allow browsing of database contents. The purpose of the OODB prototype is to research, enhance, and evaluate the OODB approach for the CLAS experiments database and other scientific databases having similar requirements.

Much work remains for the CLAS Software Collaboration. The conceptual model of a nuclear physics experiments database will likely require changes based on experience gained via the prototypes and the feedback generated from this paper.

References

[Bert91] Elisa Bertino and Lorenzo Martino, "Object-Oriented Database Management Systems: Concepts and Issues," *COMPUTER*, IEEE Computer Society, Vol. 24, No. 4, April, 1991, pp. 33-47.

[Chen76] P.P. Chen, "The Entity-relationship Model: Towards a Unified View of Data," *ACM Trans. on Database Sys.*, 1(1), 1976, pp. 1-36.

[Fren90] James C. French, Anita K. Jones, and John L. Pfaltz, eds., Scientific Database Management (Final Report), Computer Science Report No. TR-90-21, August, 1990.

[Jose91] John V. Joseph, Satish M. Thatte, Graig W. Thompson, and David L. Wells, "Object-Oriented Databases: Design and Implementation," *Proceedings of the IEEE*, Vol. 79, No. 1, January 1991, pp. 42-64.

[Khos90] Setrag Khoshafian and Razmik Abnous, *Object-Orientation Concepts, Languages, Databases, User Interfaces*, John Wiley & Sons, 1990, pp. 257-321.

[Kim90] Won Kim, "Object-Oriented Databases: Definition and Research Directions," *Transactions on Knowledge and Data Engineering*, Vol. 2, No. 3, September, 1990, pp. 327-341.

[Lagu90] "Database Systems: Achievements and Opportunities," *The "Lagunita" Report of the NSF Invitational Workshop on the Future of Database Systems Research*, Palo Alto, California, February 22-23, 1990, A. Silberschatz, M. Stonebraker, and J.D. Ullman, eds, TR-90-22.

[Ricc91] G.A. Riccardi and B.K. Ehlmann, "Object-oriented Development of Scientific Databases, an Example from Experimental Physics," *Proceedings of the First Software Engineering Research Forum*, Tampa, FL, Nov. 7-8, 1991, pp. 277-286.

[Sdmw90] *NSF Scientific Database Management Workshop*, "Individual Panel Reports and Position Papers," Computer Science Report No. TR 90-20, University of Virginia, Charlottesville, VA, March 12-13, 1990.

[Zdon90] Stanley B. Zdonik and David Maier, "Fundamentals of Object-Oriented Databases," *Readings in Object-Oriented Database Systems*, Stanley B. Zdonik and David Maier, eds., Morgan Kaufmann, 1990.