CASE DESIGN AND IMPLEMENTATION OF A GEODETIC DATABASE FOR CERN ACCELERATORS

T.K. Bhattachurjee[1], & D. Missiaen
CERN, Geneva, Switzerland

1. INTRODUCTION

The Section "Accelerator and Transfer Lines" of the CERN Applied Geodesy Group is responsible for the management of the database of survey and alignment informations for about 10000 components in approximately 50 km of particle beam lines. These data are generated from activities of different sections of the group who are responsible for different accelerator lines. These activities include aligning new components, survey of old components and checking deformation of unstable regions in accelerator lines. To accomplish these activities, classical instruments, such as levels, theodolites, distance measuring devices besides instruments developed at CERN, such as distinvar, ecartometer etc., are used.

In order to have cohesion and consistancy among different sections working under the group, it was decided to store all our data under a single database application, called "Survey". A user may access to this database through an interface called "Geode". The database was based on an empirical model and was implemented using ORACLE, the Relational DataBase Management System (RDBMS) from Oracle Corporation. Many different members of this section, most of them were CERN student trainees, have contributed in developing this application. The project commenced in 1985 and reached its current state in 1990.

This old database is being used successfully and intensively for last five years. During this period we have gained better understanding of user requirements and activities. Over the time, the volume of data has also increased and users have put forward newer requirements. With the commencement of the new LHC project of CERN, it is foreseen that the amount of data, to be handled, will see a quantum jump. But this old database is lacking a thorough documentation which is very much needed to incorporate new requirements in it and maintain it effectively. Meanwhile, ORACLE has come out with newer versions of DBMS and tools. In order to take advantages of new versions, satisfy additional requirements and to improve the response time, it was decided to redesign the whole database and develop more user-friendly user interfaces. The decision to design the model and rewrite the software using the CASE (Computer Assisted Software Engineering) Method is very much in line with the current trend in software developments. The ORACLE*Case tools were chosen, primarily, for the following reasons :

- it is an integrated structural approach, structure of the database and functionalities are visible in the form of graphical diagrams and can be modified easily;
- tables, indexes and their storage parameters can be generated and modified using simple forms;
- based on the specifications, user interfaces, such as Forms, Menus etc., can be generated;
- transition from old application to new application would become easy because of the same underlying RDBMS;
- it is having the capability to generate documentations at a reasonable level.

[1] On leave from Variable Energy Cyclotron Centre, Calcutta, India

In this paper, we shall present the step by step realisation of the new database application using ORACLE Case tools. In the next section, we briefly present the procedure followed to identify different objects and functionalities. In subsequent sections, we outline the approach for the design of the database using ORACLE*Case Designer Tools and the implementation of the new database using CASE*Generator tools. Finally, in the conclusion we will share our experiences gained during implementation.

2. IDENTIFICATION OF OBJECTS

This is a very fundamental and abstract step for the design of any application. In a database system, this step means identification of data sets, data classifications and processing routines. CASE Method suggests, to list out statements about data and functions in this step. In our case, the old database helped us to identify preliminary objects of the new Survey Database.

Primary objects which generate data and for which data are stored in the database are Survey Instruments and Accelerator Lines. Accelerator lines are composed of beam line components, such as magnets, detectors, vacuum pumps, etc. Network reference points, which are not directly related to any particular accelerator, are referenced by Areas. The broad classifications of various data, involved in 'Survey' database application, are as follows:

- Theoretical data provided by the accelerator physicists (beam orbits, voluntary displacements or bumps) or by the component designers (parameters of magnets i.e. R, S, T coordinates);
- Point reference and coordinate data (naming scheme of the points and their synonyms, computed coordinates);
- Measurement data issued from different observations (distances, levelling, angles, offsets, etc.);
- Real position data or deviations (computed);
- Calibration data (for instruments e.g. EDM, distinvar, clinometer, etc.).

The dataflow between the geodetic data gathering tools and the database is summarized in Fig. 1.

Now, we give a brief account how data in this database evolves.

The theoretical data for of Beam Coordinates are provided by physicists as ASCII files, issued from different programs (MAD, BEATCH, TRANSPORT). The mechanical parameters (position of the survey targets according to the beam position i.e. R, S, T coordinates) are acquired from component designers. The theoretical coordinates of survey targets, called socket coordinates, can thus be computed. Data for bumps are entered in the database when a component must be voluntarily displaced from the theoretical settings to change the mode of operation.

Portable computers containing the setting out data as well as the network data are used while aligning the components. All measurements, made for the alignment or during the global survey of an accelerator or for the determination of a new network reference, are also stored in these portable computers. In all cases, the *data preparation* process implements severe controls to check the validity of the measurements and formats the file of raw measurement data. *Preprocessing* programs use the calibration data to yield the input data for the *data analysis* programs, which compute either the new coordinates of the network points or deviations between a real and a theoretical position of the accelerator line components. Files containing these data and graphs showing the latter are provided to the physicists.

The special measurement data (clinometers or hydrostatic levellings) are obtained directly in PCs. Equipments are linked to these PCs via a RS232 serial line. These measurements are directly stored in the database but, as the above instruments are on test, they are not used by current analysis programs.

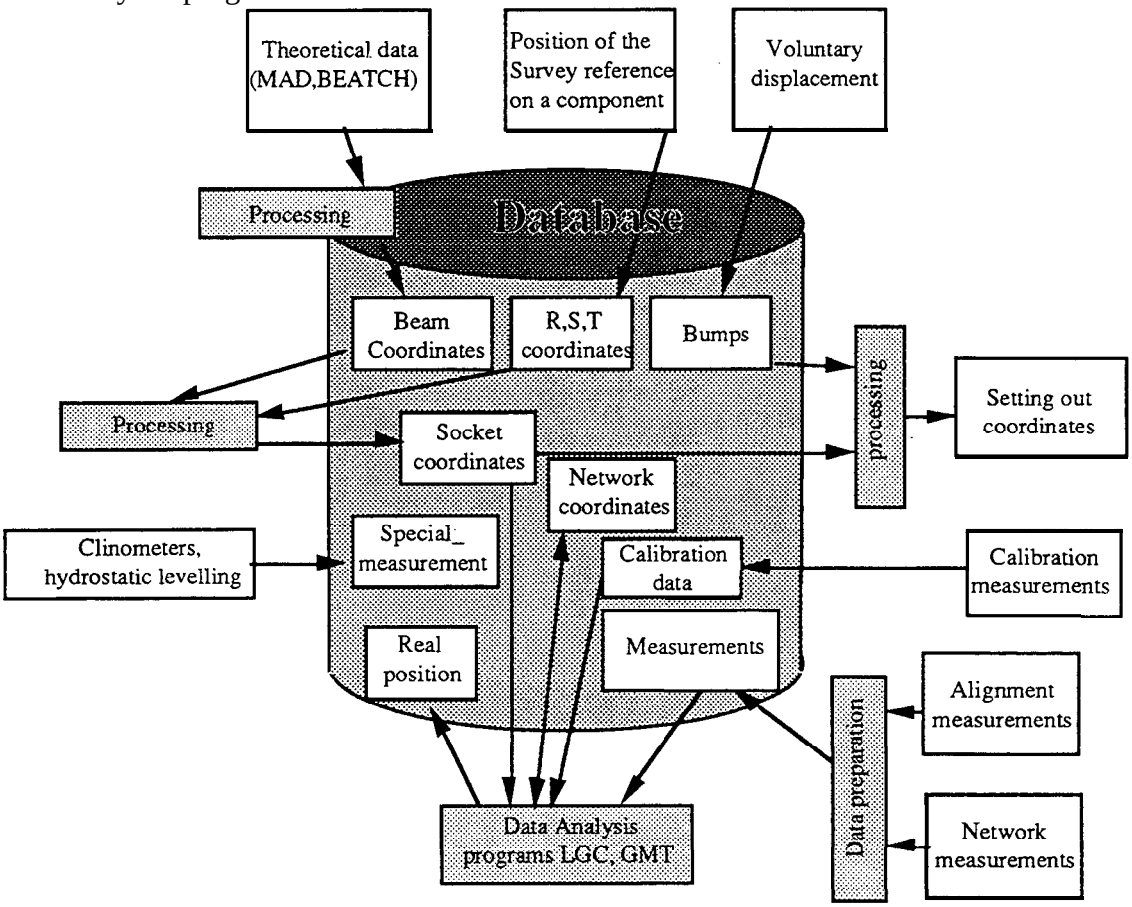


Fig. 1

3. CASE METHOD DESIGN USING ORACLE*CASE TOOLS

CASE methods are introduced to improve quality of software products by following systematic ways. It approaches methodically to split a project into manageable levels so that development can be carried out in a step by step process.

It is a proven fact that CASE encourages better quality designs, increases productivity and in turn reduces maintenance and future modification costs.

Software tools were developed to assist software engineers in solving these problems. The design informations are introduced through graphical and alphanumerical user interfaces and recorded in a central repository, a database management system. Thus, the same overall picture of the project and design data are available to all developers in a team at the same time, helping parallel developments. The same data is available throughout the production life cycle making future modifications easy.

ORACLE*Case method follows a top-down approach, where tasks are grouped in various project phases. Each phase is having a defined deliverables and has to be performed in a specific sequence. The method also includes many cross checking techniques to ensure the accuracy, consistency and completeness of the design. The phases are termed as Scoping, Strategy, Analysis, Design, (Build and Transition), Production.

The ORACLE*Case is having three major classes of tools :

- CASE*Dictionary,
- CASE*Designer,
- CASE*Generator.

The central repository, which is a multi-user database for all informations, is also called CASE*Dictionary.

The scoping of the project was very well defined, as it was decided to develop this new database system as an integrated package. It should have well documentation and user interfaces should provide sufficient help informations. As far as practicable, user environments should be kept similar to the old database so that user data acquisition softwares do not need any major modifications.

3.1 The CASE*'Designer

This tool was used during Strategy and Analysis phases. It is a graphical interface with Macintosch or MOTIF style through which the programmer enters its specifications in the repository. It includes several diagrammers, each was used at stages as explained in subsequent subsections.

3.1.1 The Entity Relationship Diagram.

This is the central part of Oracle*Case and is based on an extension of the E-R Diagram method proposed by Chen (1976). For the E-R model, entities and the relationships between entities are identified from two directional statements, such as,

- each SPATIAL POINT may be used in one or several GEODETIC MEASUREMENTs and
- each GEODETIC MEASUREMENT must be made at one or several SPATIAL POINTs.

where two entities SPATIAL POINT and GEODETIC MEASUREMENTS are involved. The E-R diagram of this relationship is shown in Fig. 2. Thus, these statements can be read from the diagram and conversely the diagram can be designed based on these statements. The subtypes of entities are identified and incorporated inside the entity blocks. For example, PILLAR, SOCKET POINT and BEAM POINT are subtypes of SPATIAL POINT. Subtypes can also have relationships among themselves. The whole E-R diagram of the SURVEY database was designed using this tool and is shown in the Appendix.

ORACLE*Case supports four kinds of relationships :

- one-to-one (1:1),
- one-to-many (1:n),
- many-to-one (n:1),
- many-to-many (n:m).

As a standard practice, many to many relationships are resolved into two many-to-one relationships by an intersection entity, as shown in the Fig. 2.

Once entities and relationships are defined, one can call *CASE*Dictionary* tools to define attributes of entities. For example, the set (id, code, comment, day, no, status, team and value) represents attributes of the entity GEODETIC MEASUREMENT. Normally, one of the attributes may be identified as an Entity key, which allows the unique identification within an

entity set. For instance, the id of GEODETIC MEASUREMENT identifies each measurement uniquely.

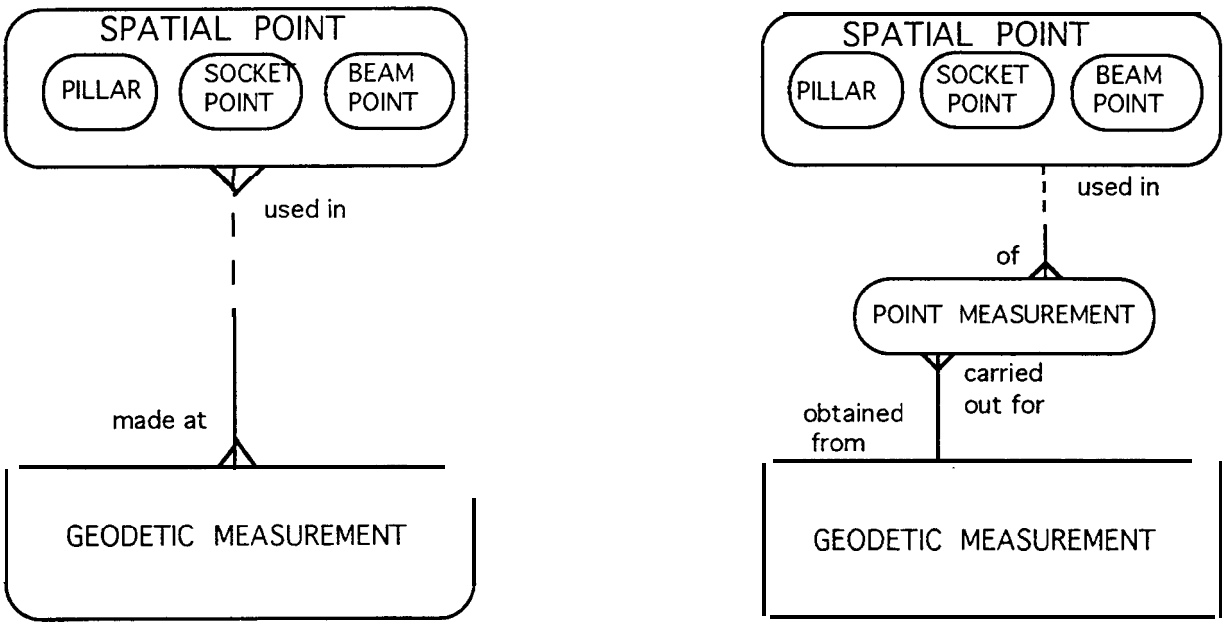


Fig. 2

3.1.2 The Function Hierarchy Diagram

The next step in the strategy step is to analyse the required functionalities. This function analysis is important as the functions are the reasons why entities are necessary.

A function describes what an action does and never indicates how the action is done. The entire business must be described as the top function which is decomposed in seven or eight major functions. Then, each function must be further decomposed until it is no more possible to decompose it. This means that when the action described by the function has started, it must always continue to conclusions. Such a (conclusion) function is called an Elementary Business Function (EBF) and later, at the design stage, each EBF will become a piece of software, i.e. a program, a menu, a subroutine, a report or simply a query statement inside a program. On the diagram, a common function can be highlighted and at the design stage such a function must become a subroutine in a library.

The complete function analysis of our application was done using this tool but is not reproduced due to lack of space.

3.1.3 The Data Flow Diagram

Once entities are defined and functional analysis done, the next step in the Strategy step is to analyse dataflow through the corresponding tool. It represents the flow of data between different functions. In this diagram, it is shown how and which entity type is the source of data and after processing this data becomes part of which entity. It is a model of how information flows within the business and across the boundary between the business and the real world. The data-flow analysis is also a cross check for functions, to show function dependencies.

3.1.4 Matrices

This is the concluding part of the strategy and Analysis stage. In this tool, a set of matrices allows to check the consistency of all analysis done so far, i.e. every function is

related to at least one entity and that each entity is used in at least one function. It is a very good way of not forgetting either entities or functions.

3.2 The Case*Generator

This tool generates tables, menus, interactives forms and reports according to the data model, the function analysis and the dataflow diagram. Unfortunately, it is not able to produce Fortran or C procedures incorporated with oracle statements. This tool is used at the Design and Build stages and produces the final product. This part is also called the implementation stage and is the subject of the next section.

4. IMPLEMENTATION OF THE SURVEY APPLICATION

The result of the strategy and analysis, using CASE method, is a data model containing about 40 entities and a function hierarchy model with three levels of decomposition and about one hundred Elementary Business Functions (EBF). In any database applications, final products- Tables, Views, Indexes, Clusters are required for data storage while Forms, Menus and High Level Language (HLL) procedures act as user interface to interact with the data. In this section, we shall present our approach in achieving these products using ORACLE*Case tools.

4.1 Tables, Indexes and Clusters

Normally, all the entities do not become tables and the most common example is when an entity contains several subtypes, one has the choice to create one single table (based on the super-type entity) with a column to distinguish every subtype or as many tables as subtypes. In all these situations, it was decided to create one single table for the following reasons :

- the maintenance of the unique key is easy,
- reduced proliferation of tables,
- all the data with same attributes would be in the same table,
- ensures the intended use of the data.

Out of 40 entities of the data model, only 20 have become tables.

Once decision has been taken about entities which will become table, CASE*Generator is used to generate them 100% automatically. In tables, columns are generated from attributes of the entity and its relationships with others.

CASE*Generator also creates Indexes and Clusters automatically once specified. Unique Indexes are always created for attributes which plays the roles of the primary or the unique key to ensure integrity of uniqueness. Additional non unique indexes and Clusters are created to increase response time. Since, our database application is still at the Build stage of CASE method, this part of the implementation is not yet completed.

4.2 Views

A view in a relational database system is a way to display a part of a table, to mask certain columns and to realise a join or an intersection of several tables. The data are not physically stored in a view but the data of the underlying tables are accessed when the view is accessed. Unfortunately, CASE*Generator is not capable to generate views based on the structures of the model.

In our application, views are mainly used to enforce security constraints in the database. For example, for a user with a restricted use of a table, a view is created and the user is provided with a synonym of the view, same as the table.

4.3 Application Softwares

A database application can be considered to be widely usable, if it is accessible by a user without any knowledge about the internal database storage structure or any programming language. Application softwares hide the complexity of a database structure and provide the required interactive interface between the users and the database. In a ORACLE database application, these softwares constitute SQL*Forms, SQL*Menus, PRO*C or PRO*Fortran procedures. Using ORACLE*Case tools, the components of a software package are defined as modules which are formed by groupings of the functions discovered by the function analysis. In the SURVEY application, we have identified about 70 modules. A module is classified as a Menu, Form or Utility, based on the role of the corresponding function,.

Menus, in our application, connect different pieces of softwares of widely different origins. The CASE*Generator generates these menus from the modules typed as Menu to the extent of 100%. Forms are the most convenient interface for users to Query, Insert or Update the database. CASE*Generator is very efficient in producing simple forms but most of our forms were rather complex, as they required trigonometric computations. Some of these forms could only be created up to 30% using the Generator while 70% are to be developed using usual Oracle tool.

Among Utility softwares, functions of SQL*Loader are used to insert formatted ASCII files directly into the database. For example, the insertion of theoretical data coming from physicists and component designers are entered using SQL*Loader functions in the modules.

A large number of utility softwares deal with data from files. These data need partial processing before insertion in the database. The modules dealing with such data are written completely manually using PRO*Fortran or PRO*C to incorporate SQL statements for database access. Some of these procedures are utilised by Forms in the form of subroutine calls, termed as User-exits. In fact, a major part of our activities are directed in developing these packages.

The database and the application software package are located on a VAX machine from DEC. Users can run these softwares from their own user accounts concurrently. Thus all the data and softwares remain unique to everybody.

5. CONCLUSION

This new database system and most of the application softwares have been released to users at the beginning of August 1993 and are performing satisfactorily.

THE CASE BENEFITS

The top-down method is new in the CERN environment. Precedently, developers used to begin writting codes without having a global overview of the project. This method enforces, a developer to spend sufficient time on the strategy and the analysis at the beginning of the project. The data model, designed using CASE tool, is more consistent and logical than the one created empirically. Tables and constraints are defined with a better quality and the structure is ready to accept Oracle V7.

Application software interfaces to the database have been produced, on an average 40% automatically, using common templates provided by the generator so that it is very homogeneous and rigourous. Beginning in October 92, the application has been developed in

about ten months by one of us working at 100% and the other working at 50% of time. As a comparison, the old system was developed in five years by two student members on an average.

The fact, that one of the developer is also one of the user, is a handicap because he may have an overview of a solution of a problem which is not the best one. The arrival of one of us (TKB), a non topographic person, in this project was very positive. As he did not know everything of the project, he was obliged to interview the project leader and end users to identify new approaches to problems.

CASE tools generate on line prompts at every steps automatically. It also generates documentations for presentations and consultations. Tools are also provided with on line helps at each steps.

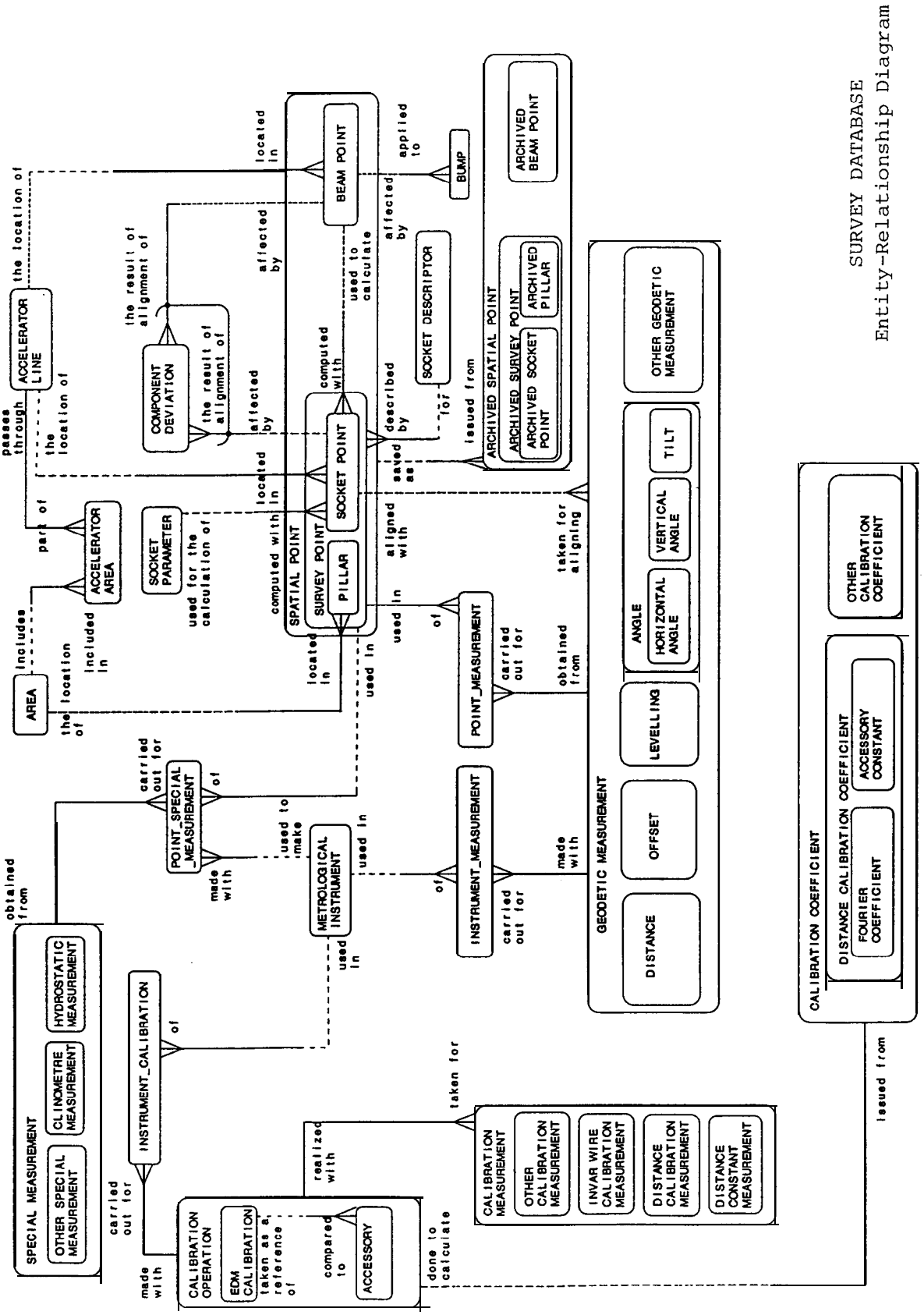
As the model definitions are stored in the dictionary, the future modifications will always be possible and can be developed quickly, using integrated packages of ORACLE*Case.

THE CASE DISADVANTAGES

It is a rather heavy method to start up and needs considerable time to be understood by a developer.

BIBLIOGRAPHY

R. BARKER : CASE*Method
A. DANEELS : CASE in CERN's Accelerator Sector
D. MISSIAEN: The setting up of a Database for a large Engineering project


SURVEY DATABASE
Entity-Relationship Diagram
SPECIAL MEASUREMENT
OTHER SPECIAL MEASUREMENT
CLINOMETRE MEASUREMENT
HYDROSTATIC MEASUREMENT
INSTRUMENT_CALIBRATION
CALIBRATION OPERATION
EDM CALIBRATION
ACCESSORY
CALIBRATION MEASUREMENT
OTHER CALIBRATION MEASUREMENT
INVAR WIRE CALIBRATION MEASUREMENT
DISTANCE CALIBRATION MEASUREMENT
DISTANCE CONSTANT MEASUREMENT
POINT_SPECIAL MEASUREMENT
METROLOGICAL INSTRUMENT
INSTRUMENT_MEASUREMENT
POINT_MEASUREMENT
AREA
ACCELERATOR AREA
ACCELERATOR LINE
SOCKET PARAMETER
SPATIAL POINT
SURVEY POINT
PILLAR
SOCKET POINT
COMPONENT DEVIATION
BEAM POINT
SOCKET DESCRIPTOR
BUMP
ARCHIVED SPATIAL POINT
ARCHIVED SURVEY POINT
ARCHIVED SOCKET POINT
ARCHIVED PILLAR
ARCHIVED BEAM POINT
GEODETIC MEASUREMENT
DISTANCE
OFFSET
LEVELLING
ANGLE
HORIZONTAL ANGLE
VERTICAL ANGLE
TILT
OTHER GEODETIC MEASUREMENT
CALIBRATION COEFFICIENT
DISTANCE CALIBRATION COEFFICIENT
FOURIER COEFFICIENT
ACCESSORY CONSTANT
OTHER CALIBRATION COEFFICIENT
carried out for
made with
obtained from
taken for
used to make
used in
realized with
compared to
done to calculate
issued from
of
part of
includes
included in
the location of
passes through
the result of alignment of
affected by
used for the calculation of
computed with
used to calculate
located in
aligned with
saved as
described by
for
applied to
obtained from
taken for aligning