User Oriented Design Methodologies and the Possibilities of Establishing International Databases Dealing with the Effects of Pollution

Abstract

Methodological and modelling issues concerning information systems dealing with pollution data are discussed. A general analysis of the main objectives that must be followed when designing such systems is given and the concept of a database is introduced. Emphasis is given to the fact that a database solution is composed of methodologies, models and technologies. Modern methodologies individuate three important steps in information system design : the conceptual phase, the logical phase and the physical phase. The first phase, in particular, is of interest in the present context, because in this phase a high degree of involvement is required not only of the technicians but also of the experts in the problem. In the conceptual design phase widely used conceptual models are introduced for representing both data and functions. Finally, a brief review of the main functions of CASE systems is presented.

MOTIVATION

In this paper, we tackle the problem of identifying effective methodologies, models and technologies for representing, designing and developing information systems dealing with pollution data. In general, pollution data can be of interest in many different contexts and for different purposes. The scientific community is interested in studying the various phenomena associated with pollution and establishing models that can describe them in the desired degree of detail. These theoretical models should be in agreement with experimental results and can make use of experimental measurements for starting and monitoring their calculations. The public administrative sector monitors pollution data in order to plan actions regarding different aspects of public interst (transport infrastructures, conservation of the cultural heritage, development of urban and suburban areas, etc.). The healthcare

field, in particular, is highly interested in studying and monitoring pollution data, both for research and practical purposes. These studies impose new specifications and constraints on the information systems that must support them. Therefore if, for example, as part of a healthcare project, studies on epidemiology are to be carried out, it is necessary that the information system is able to organize historical data about pollution and make them available in as quicly as possible.

It is clear, therefore, that an information system managing pollution data must be able to:

- Manipulate rapidly large volumes of data on a variety of pollution phenomena. Large quantities of data must be accepted, checked and made available for variety purposes.
- Organize such data in a rational, reliable and uniform way. The organization of the data must reflect their intrinsic characteristics, independent of the various applications to which they may be put.
- Present the same classes of data in different forms (graphs and illustrations, maps, analyses, etc.). Pollution data are of interest to different users and organizations. Therefore it is necessary to provide the system with many different alternative representation forms, that allow a wide 'communication bandwidth' between the system and its user.
- Make these data available for different types of application. The data management system must be an independent part of the whole information system, exchanging information with other modules by means of a well defined interface language. In this way, it is easy to give support to new applications that require access to pollution data, providing them with the correct interface for speaking with the data management system.

A NON-INTEGRATED SOLUTION

In the early days of Information Technology, a not-integrated approach to the design of an information system was often followed (Fig. 1).

Using this approach, every specific application that requires access to pollution data defines and manages its own set of archives, that have structures that reflect the particular needs of that application. This approach has several shortcomings:

- Information is fragmented, being split up between several archives. The whole information resource is not available in a unique, integrated set of archives. Every archive has its own structure and organization and must be managed according to particular rules that depend on the application in question. This situation leads to difficulties when we wish to use the same data for purposes other than the original ones.
- Fragmentation leads to duplication of effort and unnecessary costs for the information management. The basic functions for managing data are

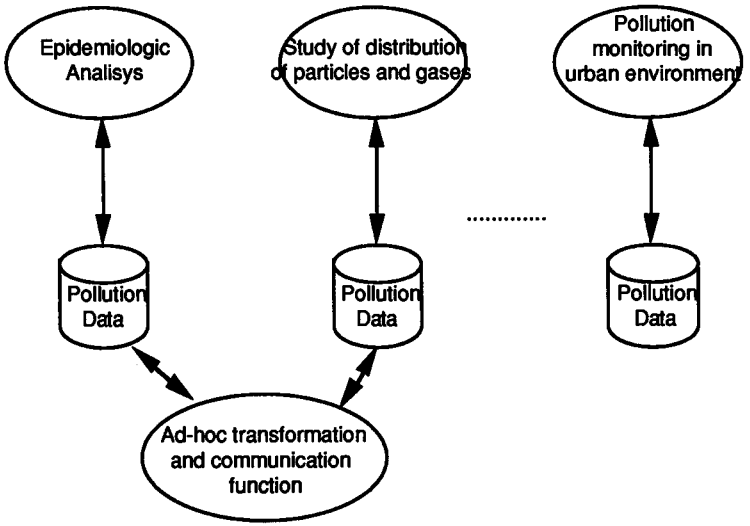

Fig. 1. A non-integrated approach to the design of an information system.

always the same. Therefore, if we do not have a unique, integrated database, we have to repeat the design and development of these functions for every set of archives, thus incurring unnecessary effort and cost.

- Lack of reliability, due to (possible) multiple sources for the same information and inconsistencies in the data. If we do not have an integrated design for the data in question, it is possible that the same datum is observed more than once, leading to redundancies and inconsistencies (the same datum can be accepted by different people in different places and at different times, leading to different values).
- Difficulties in communicating between systems, due to different models and formats for the data in question. Every set of archives is designed to match the requirements of the relevant application. This approach can result, for example, in different formats for the same information in different archives, neccessitating the establishment of transformation policies and rules for exchanging information between systems. A more conceptual problem arises when different models are used for representing the same data in different systems. In this case, it is necessary to establish transformation rules between the representation constructions of the models.

A DATABASE AS A SOLUTION

In order to overcome the shortcomings outlined above, it is necessary to adopt a different approach, what we call the 'database approach'. This approach comprises three aspects :

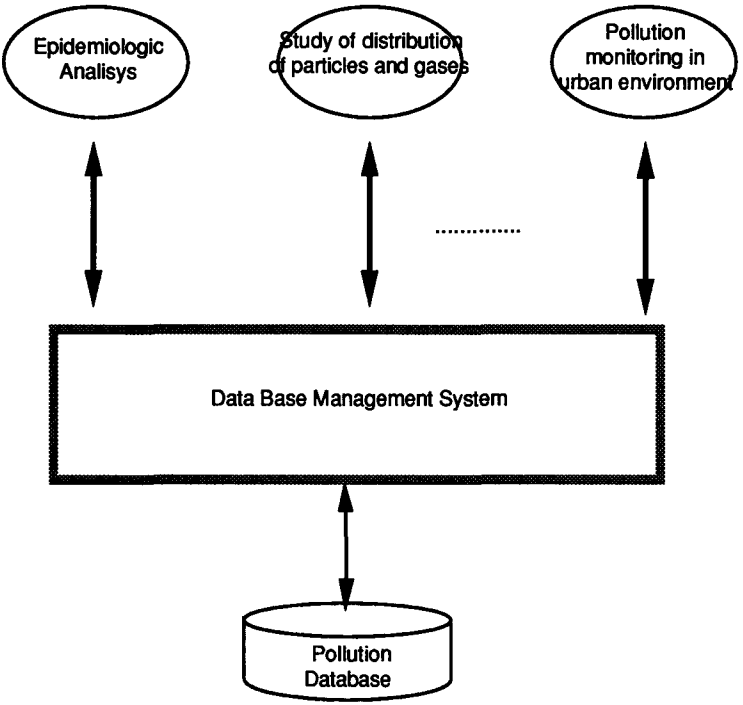

Fig. 2. Architecture of a system based on database technology.

- A design methodology: the adoption of this approach implies the use of a well-defined methodology that guides the designer to obtain a realistic picture of the information system in question.
- Models for describing various classes of data: the methodology suggests which models should be adopted during the various design activities, and how they should be used. Several models must be available, because many different activities are to be performed and different classes of data are to be described. Therefore, for every methodological step, one or more models must be suggested.
- A technological framework: once the designer has produced the abstract representation of the system, in terms of design models, it is necessary to transform it into a running system. Therefore, a powerful and flexible development environment must be available.

The key concept behind the database approach is to produce a unique, integrated representation of the data in question, independent of the functions of which the system is composed. This integrated representation is managed by a software module, called a DataBase Management System (DBMS), which offers generalized services through a well-defined interface. The proposed architecture is depicted in Fig. 2.

This approach offers the following advantages:

- An integrated and homogeneous environment for the definition of data: all the data of interest are defined in an integrated representation, using a unique model. This avoids inconsistencies and (if so desired) redundancies. The meaning of the various classes of data is clear and is given independently of the functions used to gain access to them.
- Centralization of management functions: the basic management functions are provided by the DBMS. In this way multiple efforts for developing them and possible inconsistencies between different management policies are avoided.
- Centralized control over integrity constraints and rules: it is possible to define the integrity constraints for the data of interest, leaving the commitment to manage them to the DBMS. In this way, the applications need not be involved with them. Such an organization produces a faster and more flexible system.
- Support for multi-user environments: DBMSs provide processing with transactions. This capability allows many users to access the database at the same time without loosing data consistency.
- Automatic recovery: once a system failure has occurred (in hardware, software or both), the DBMS is able to rebuild the last consistent status of the database by means of a transaction log which it maintains automatically. This feature avoids the loss of important data.
- Continuous evolution support, both for data and functions: database technology allows the DBMS to be decoupled from the other software modules. Therefore, modifications in data representation and integrity rules, resulting from modifications in the system requirements, only take effect inside the DBMS and do not affect the applications. This organization allows the set of data in question to be modified and extended more freely and at less cost. Moreover, new applications can be freely added to the system, without affecting the existing modules.

Conceptual tools

Database technology is not sufficient without a good design methodology. Indeed, the available technology exhibits the following problems:

- Dependency of the hardware/software platform: every DBMS is available on a number of machines and uses specific operating systems, programming languages and tools. Therefore, designing an information system on a specific DBMS is not reccommended.
- Existence of different database models: every DBMS makes reference

to a 'logical database model'. The most important models described in the literature are the hierarchical model, the network model and the relational model (Codd, 1970, Ullmann, 1982). If the data design references to a particular DBMS it is difficult to translate it to another one.

- Logical models are not user-friendly: the available logical data models adopt concepts and notations that are far removed from the ultimate users perspective and knowledge. In this case the user can scarcely understand (and certificate) the logical representation of the data.

These considerations show that it is essential to base the first design phase of the information system on a set of 'conceptual tools', capable of representing the real point of interest in an understandable and technologically independent way. In terms of systems managing pollution data these considerations are particularly significant because the design of such systems requires putting together knowledge and experience from many different fields and topics. Therefore, the conceptual tools adopted for describing data and functions must be understandable by both experts in the field of research and technicians. The conceptual tools allow a conceptual representation of the system, formed by one or more conceptual schemes, that can be validated by the users and also used by technicians as a starting point for the physical design.

The methodological approach

Pollution is often a wide-scale phenomenon, involving not only different sites within the same country, but also different countries and nations. This is because of the large variety of its implications which impose long-term political approaches at the international level. In this context, different interests and projects, possibly in different countries, need to describe data in a standard way in order to allow:

- Data exchange between different applications.
- Comparability of results obtained by means of different systems.
- Integration of different applications into larger systems.

It would, therefore, be extremely useful to establish standard guidelines for methodologies and models being adopted. Nowadays, it is generally accepted (Batini *et al.*, 1988), that the overall software design activity must be divided into three basical steps: the conceptual design, the logical design and the physical design (Fig. 3). Firstly, it is important to identify a way of describing the various components of the system (requirements, organization, data and processes) without the need of adopting languages and terminologies related to specific technological aspects. This task is accomplished in the conceptual design phase, which represents the initial step of the project.

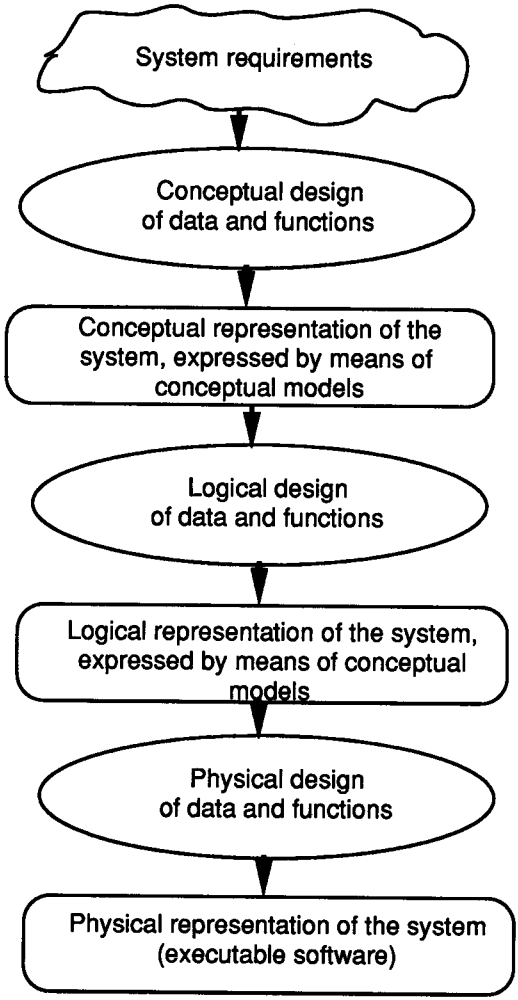

Fig. 3. Conceptual, logical and physical design phases.

The conceptual design activity, therefore, aims at describing the main components of an Information System (i. e. data and functions) by means of representations independent of the actual technology that will be adopted for its implementation. During this phase designers and users work together to obtain representations of data and functions that meet the system requirements. So, it is necessary to adopt, for this phase, conceptual models that can be understood by non technical people and yet formal enough for representing the characteristics of the system in an unambiguous way.

The logical design phase allows the designer to define system characteristics which are dependent on the classes of technology chosen. During this phase logical models for data and functions are used and the designer

is asked to perform model transformations in order to derive the logical representations, from the conceptual ones. For example, if the designer decides to adopt a relational DBMS with its fourth generation language, he/she has to adopt, as logical models, the relational model for data and a fourth generation language specification.

The physical design phase allows the designer to individualise all the physical characteristics of the system which are dependent on the actual technology (DBMSs, programming languages, OS features, etc.) which will be adopted for the implementation. This phase also comprises a transformation of logical representations into physical ones, followed by refinements for using the development environment in the best way. It is important to emphasize that the various steps must be integrated in a consistent framework, proceeding on the basis of the mutual results. This means that the output of one phase must represent the basic input for the subsequent one.

Conceptual models

Let us focus our attention on the conceptual design phase and on the models that can be used during it. We can divide the available conceptual models into two categories:

- Conceptual models for data: this category allows the representation of data at the conceptual level. We distinguish, in particular, between conceptual models for elementary data (data representing phenomena in an analytical way, with the maximum degree of detail) and conceptual models for aggregated data (data representing phenomena in a synthetic way).
- Conceptual models for functions: this class of models allows representation, at the conceptual level, of all the functions that compose the system, and their interactions.

Conceptual models for elementary data

A conceptual model for elementary data allows the basic classes of data in question, their characteristics and relationships, to be represented, in a formal and yet simple way. A widely used conceptual model is the Entity-Relationship (ER) model (Chen, 1976). This model provides the following constrictions:

- Entity: represents a homogeneous set of relevant concepts. In an air-pollution monitoring system, for example, we can have entities such as device, place, physical/chemical property, etc.

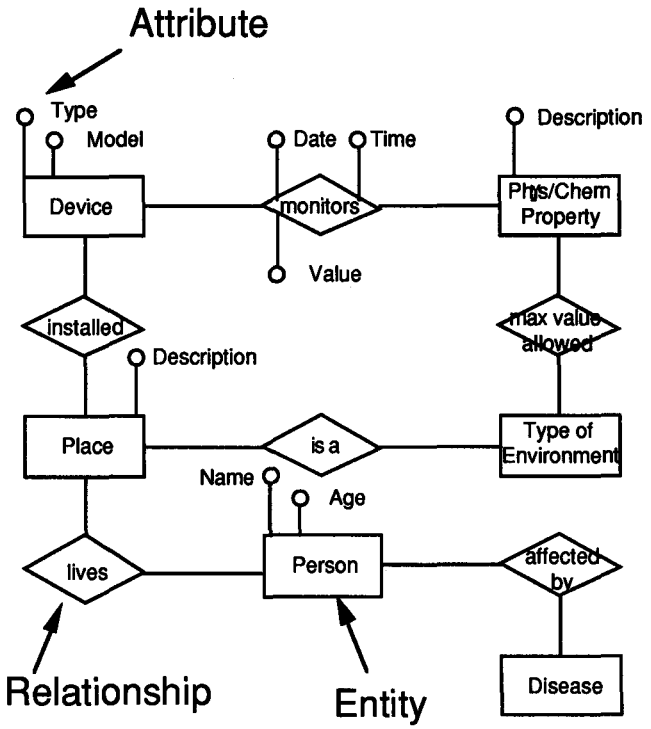

Fig. 4. ER Scheme of the air pollution monitoring system.

- Relationship : represents a conceptual link between two or more entities. For instance, in the example above, we can install a relationship that connects device and place.
- Attribute : is a characteristic of an entity or relationship. For example, device has the attribute type, which describes the particular kind of device we are dealing with.

The ER model is provided with a graphical notation, that allows ER schemes to be drawn. An example of an ER scheme is shown in Fig. 4.

Conceptual models for functions

A conceptual model for describing functions allows the designer to represent the relevant functions in a formal way, independent of the final hardware/software platform. A popular model for performing this task is the Data-Flow (DF) model (De Marco, 1978, Gane and Sarson, 1979). This gives the following constrictios :

- Process : is an elementary or complex task performed by the system. For example, the acquisition of input data by means of a monitoring device, the simulation of a complex phenomenon using a mathematical model.

- Data Store : is a repository of homogeneous data, accessed and managed by the processes. For example, data about gases, provided by a set of sensors.
- Interface : is an active object, with which the system exchanges information. For example, an external organization interested in pollution data.
- Data Flow : is a flow of information between processes, interfaces and data stores.

The DF model also has a graphical representation, that allows DF schemes to be represented graphically.

Conceptual models for aggregated data

The ER model is not expressive enough to represent all the kinds of data relevant to pollution problems. Another important class of data is encountered when we attempt to aggregate primary data in order to obtain a syntethic representation of the phenomena. Let us suppose, for example, that we want to answer the following question (with reference to the ER scheme shown in Fig. 4). « Give me the correlation between the incidence of a particular disease in people over 18 years in age and the average concentration value of a particular gas in the air. » The answer to such a question can be obtained, with some degree of difficulty, directly from the ER scheme, but it is noticeable that no aggregate concept can appear in the ER scheme without enforcing the use of the model. It is, therefore, necessary to adopt another model, specifically designed for aggregate information. Such a model is the Statistical Conceptual Model (SCM) (Batini and Di Battista, 1988). In this model, the main available constructions are :

- Object Class : a collection of objects relevant to the analysis, which can be viewed as obtained from one or more entities by means of a query.
- Classification : a partition of a class of objects, with respect to some characteristics.
- Class of Data : this is obtained from a classification or another class of data by means of a statistical operator (average, sum, count, etc.).

This model also has a graphical notation that allows statistical schemes to be represented graphically. In Fig. 5 the SCM scheme corresponding to the above query is illustrated.

A METHODOLOGY FOR THE CONCEPTUAL DESIGN PHASE

A conceptual design methodology must allow the usage of different conceptual models in an integrated way, in order to ensure consistency in the different representations of the reality being considered. In particular, we observe that :

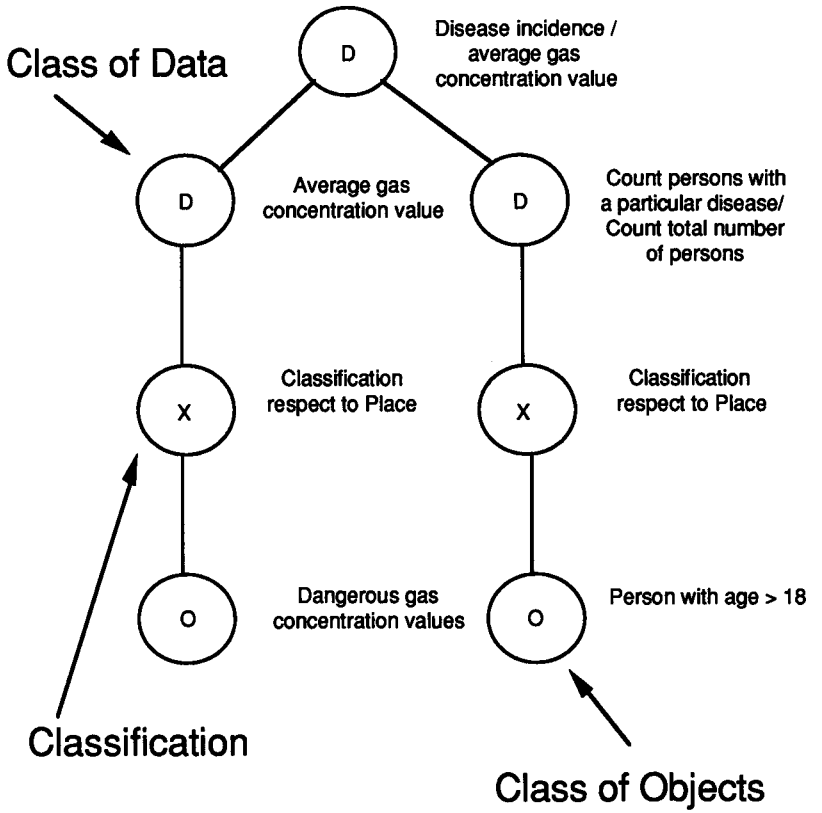

Fig. 5. SCM Scheme representing the sample query.

- The representations, obtained by means of ER and DF schemes, are complementary, so it is necessary to obtain mutual consistency and completeness.
- The representation of aggregate data, by means of SCM schemes, is based on elementary data modelled by means of ER schemes. Therefore, ER schemes should be complete and consistent with respect to SCM schemes.

The main activities performed during the conceptual phase are depicted in Fig. 6. They can be summarized in an initial definition (of elementary data and functions) and a series of incremental refinements until the scheme completely satisfies the needs. Ultimately aggregate data are designed, possibly producing new requirements of both the elementary data and functions.

The design process can be described in more detail as follows.

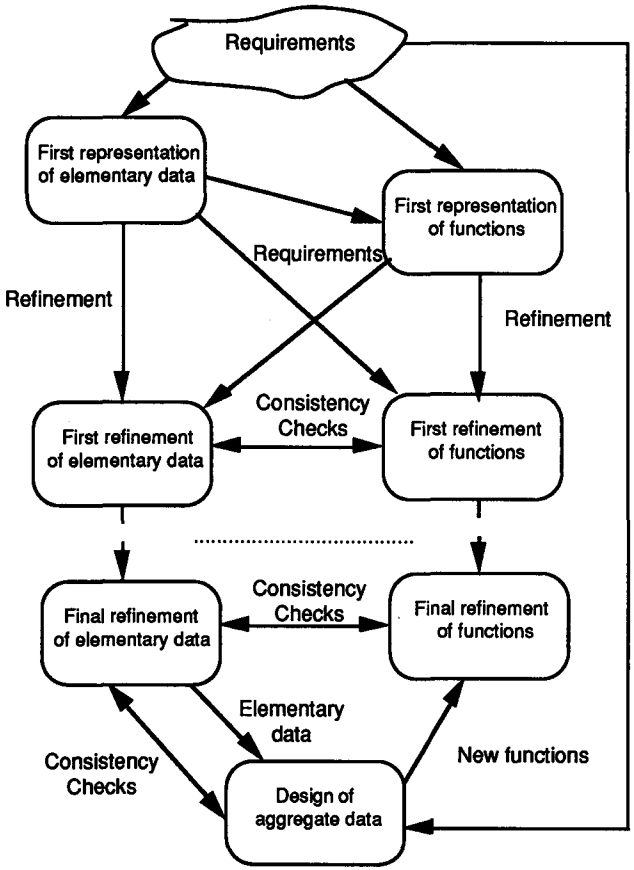

Fig. 6. A methodology for the conceptual design.

Starting from the system requirements, the working group obtains a first, very abstract, representation of elementary data, by means of the ER model. This first representation is often called the 'skeleton data scheme'. This first representation of the data, together with the functional requirements, allows the working group to individuate a first, very abstract, representation of functions, by means of the data-flow model. This diagram is generally called a 'Context Diagram', because it is often composed of one process representing the whole system, which interacts with several interfaces, representing bodies external to the system.

It should be noted that, normally, it is desireable to start with the conceptual design of the data, because the datum source is more stable than the set of functions relevant to its organization. For the same reason, it is an important methodological guide-line to drive the whole conceptual

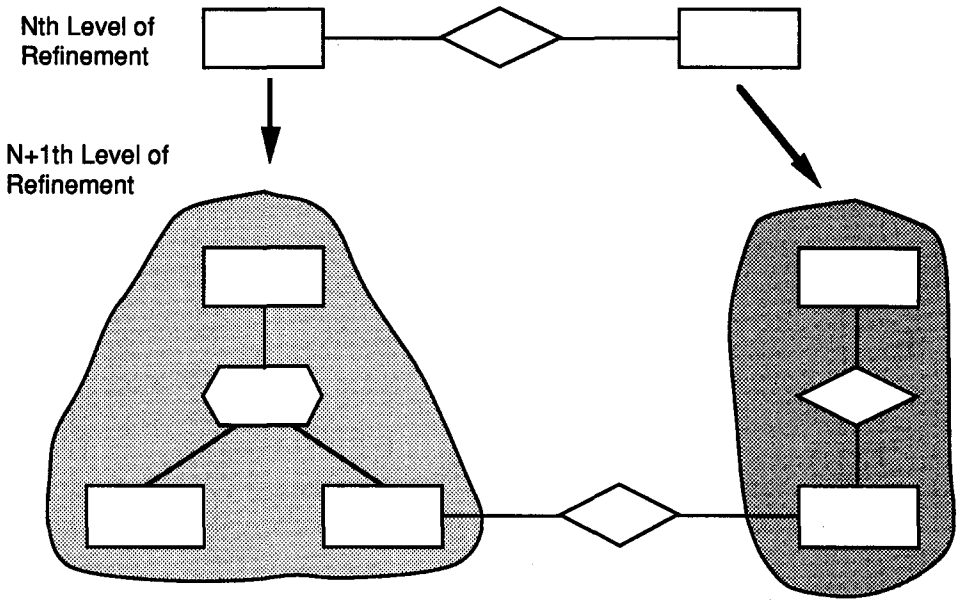

Fig. 7. Separation among Concepts, at Different Levels of Abstraction.

design phase on the basis of the data requirements rather than on the functional ones. Once the first representations of data and functions are obtained, they must be refined, in order to pass from very abstract and complex concepts to more simple ones. Starting from data and function representations at one level of abstraction, the refinement process allows the identification of new schemes which are more detailed at a lower level. The refinement mechanisms allow the transformation of a concept in a subscheme, i.e. a structured set of concepts at a lower level of abstraction. It is difficult, if not impossible, to give formal rules for performing the refinement process in the correct way, because this activity is highly semantic-dependent. However, two general rules can be given :

- Two distinct concepts, at one level of abstraction, should also remain distinct at lower levels of abstraction. This means that the subschemes obtained by refinement from the abstract concepts should have few or no concepts in common (Fig. 7).
- If we have one concept, at one level of abstraction, connected to other concepts by means of various constrictions, the corresponding subschemes, at a lower level, should be connected by means of a similar set of constrictions (Fig. 8).

If these two rules are followed in the refinement process, it is possible to be more confident of the fact that the various data and function schemes represent the same reality of interest, at different levels of abstraction.

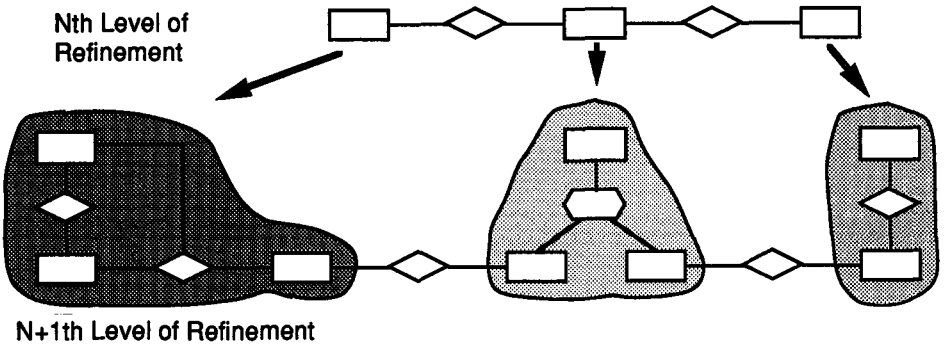

Fig. 8. Connection among concepts, at different levels of abstraction.

At every level of abstraction, data and function representation should be mutually complete and consistent. An ER scheme is complete with respect to a DF scheme if, for every data store present in the DF scheme, there is an ER subscheme which has the same information content. When this kind of completeness is achieved, the ER scheme represents a database capable of poviding the information requested by the system represented by the DF scheme. *Vice versa*, a DF scheme is complete with respect to an ER scheme if all the information represented in the ER scheme is present in some data store in the DF scheme.

When this kind of completeness occurs, the DF scheme represents a system capable of accessing the whole of the information stored in the database represented by the ER scheme.

Consistency problems occur when the same concepts are represented in different ways in the ER and DF schemes, or when different concepts are referred to by the same name. It is necessary, in these cases, to remove such contradictory situations. Once the most detailed representations of data and functions have been obtained, it is necessary to design the statistical data by means of the CS model. Such data must be defined on the basis of existing elementary data, so as to ensure consistency and completeness between ER and CS schemes. Moreover, statistical data probably need their own management functions. Therefore, the definition of aggregated data allows an enrichment of the final DF schema.

At the end of the conceptual design process, the result is a structured set of ER and DF schemes, ranging from the highest level of abstraction to the maximum level of detail. Each scheme is clearly related to its parent and son schemea, refers to the same model, and meets the mutual completeness and consistency requirements of one scheme expressed by means of the other model. The final result also comprises CS schemes which represent the aggegated data of relevant to the system. These schemes, together with the final ER and DF schemes, are mutually complete and

consistent, and, therefore, provide a complete and consistent conceptual design of the system.

CASE TOOLS

The methodologies and models described above are required to perform a variety of activities, some of which can be boring, repetitive and mostly syntactical. It is, therefore, very useful to have automatic tools available that can support the designer during the various steps of the methodology (McClure, 1989). Computer Aided Software Engineering (CASE) tools can help people to design and develop applications. CASE supports in :

- describing the various aspects of the reality in question by means of schemes : in this context, CASE tools provide graphical editors that allow the designer to specify the correct schemes in a graphical way and then to store the information in a project database :
- checking the completeness and consistency of the set of schemes obtained so far : CASE tools provide checking algorithms that access the project database and detect situations of possible conflict.
- transforming the conceptual description of the system into a logical/physical one : once the conceptual schemes are obtained and validated, it is necessary to transform such conceptual representations into machine-dependent ones. These logical/physical representations are the basis for the subsequent development. Most of the effort needed for transforming conceptual schemes into logical/physical ones can be performed by the CASE tool by means of well established transformation algorithms.
- producing a complete documentation of the system : the CASE tool can access the project database in order to produce a detailed documentation of all the parts of the information system. It is clear, therefore, that CASE technology can be of real help in designing complex information systems, like those of interest in this paper.

Enrico MELIS
Gesi s.r.l.
Via Rodi 38
I - 00195 ROMA, Italy

REFERENCES

BATINI, C., CERI, S. and NAVATHE, S.B., 1988, *Logical Database Design using the Entity-Relationship Approach*, Benjamin & Cummings.

BATINI, C. and DI BATTISTA, G., 1988, *Design of Statistical Databases : a Methodology for the Conceptual Step*, in *Information Systems*, 13(4).

CHEN, P.P., 1976, *The Entity-Relationship Model : Toward a Unified View of Data*, in *ACM Transactions on Data Base Systems*.

CODD, E.F., 1970, *A Relational Model for Large Shared Data Banks*, in *Communication of the ACM*, vol. 13, p. 377-387.

DE MARCO, T., 1978, *Structured Analysis and System Specification*, Yourdon Press/Prentice Hall.

GANE, C. and SARSON, T., 1979, *Structured Systems Analysis: Tools and Techniques*, Prentice Hall.

MCCLURE, C., 1989, *CASE is Software Automation*, Prentice Hall.

ULLMANN, J.D., 1982, *Principles of Database Systems*, in *Computer Science Press*, 26.