From a "well designed" database to AD/Cycle tools: a reengineering experience*

Oreste Signore - Francesco Celiano

CNUCE - CNR - via S. Maria 36 - 56126 Pisa (Italy)

Abstract

An application for the management of users and resources of a general purpose data processing center, previously developed using a manual database design methodolgy based on enriched entity relationship approach, has been redesigned using a CASE tool of the AD/Cycle "family".
The existing application was based on a hierarchical DBMS (SYSTEM 2000) and was written in third generation language.
The new application uses a relational DBMS (SQL/DS) and programs are written in a fourth generation language.
During the reengineering phase, the conceptual schema has been slightly modified, according to new requirements and constraints; in this phase users and application programmers have been directly involved in the use of the tool.
The development times, accuracy, user involvement, documentation quality, CASE tools benefits and drawbacks are discussed and compared.

Introduction

CNUCE is an Institute of CNR, whose aim is not only to develop researches in the computer science area, but also to supply the scientific community with computing power and resources. From this point of view, the management of user requests takes a peculiar importance, as the institute is faced with a wide variety of user requests, on a non scheduled basis.

On the basis of these considerations, in 1983 the decision was taken to razionalize the user resource management, and therefore to substitute the previous approach, based on a variety of files and programs, with a more comprehensive and integrated database approach.

To be more specific, a careful analysis of the activities performed lead to the identification of four main areas:

- Teleprocessing area (network elements management, planning of interventions, auditing and prevision of costs).
- User management area (access to computer resources, accounting, etc.).
- Hardware and software resources (management of acquisitions, contracts, etc.).
- General services (typography, inventory, etc.)

* This work has been partially supported by "Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo" of C.N.R..

In each area, some functions were performed manually, other were automated. In any case, the degree of integration was very poor.
The attention was focused on the *User management area*.

The application: an informal specification

Following functions to be performed can be distinguished in the user management area:

- Management of the library (sale of the manuals and other documentation).
- Management of user access accounting codes and registration of people allowed to get access to the resources.
- Management of resources (virtual machines, tapes, disks, etc.), that is, linking of resources to the appropriate user accounting codes, and recording relevant information about every transaction.
- Management of invoices (every resource must be accounted to the customer).
- Evaluation of the resource usage in order to plan the development of the computer center.

Every user, in order to gain access to the CNUCE computer resources, must sign an agreement, and get one or more user codes. For each user code, a representative is identified, who can ask for every resource the code is allowed to ask for, and can grant this permission to other persons.
Each request by the user can modify the number and the kind of allocated resources. In addition, even if a code is canceled, information must be kept alive for a minimum of two years (for accounting and possible claims), even if resources may be allocated to other customers.

The manual methodology approach

The DATAID methodology

The database was designed using the DATAID methodology [Ceri83], defined by several groups in the context of the "*Progetto Finalizzato Informatica*", a 5-year research project financed by the National Research Council of Italy. At the time being, the DATAID methodology was essentially a manual methodology, even if some components were automated, and it was conceived as a methodology for database design. For this reason, it was supposing the analysis of the enterprise as already done, and it was concentrating on tools for supporting the sequence of the phases for database design:

- user requirement collection and analysis;
- conceptual design;
- logical design;

- physical design.

The *user requirements collection and analysis* phase was getting information from user interviews, and collecting into glossaries (Data Glosssary, Operations Glossary, Event Glossary) and forms. Constraint requirements were also collected.

The *conceptual design* phase was based on the process of views integration, where a conceptual view is defined as the formalized representation of both the statical (data) and the dynamical (operations, events) requirements of a database application, which are relevant to one environment of the organization.
The data model used is an extension of the Entity-Relationship model proposed by P. Chen ([Chen76]). These extensions are:

- *Abstraction hierarchies*: possibility of defining subsets, partitions or restrictions of entity sets ;
- *Aggregates*: set of attributes that can be referred as a single property;
- *Repeating attributes*: multivalued attributes;
- *Identifiers*: attribute or group of attributes that uniquely determines an entity. Entities may also be identified through other entities associated with them (external identifier).

Dynamic aspects are represented by means of Petri nets.

After all, the DATAID methodology is basically data oriented, and, because of this, procedural aspects are not emphasized. Anyway, procedural aspects are taken into account, to some extent, by one of the tools, namely Galileo language.

Specific tools, oriented toward specific DBMS families, were developed for logical and physical database design .

The database

The application for user management was developed following the specifications of the DATAID methodology. The conceptual schema was making extensive usage of IS-A hierarchies and multiple attributes.

The adopted DBMS was SYSTEM 2000, and mapping on this specific DBMS was performed defining rules tailored to it.
The resulting database logical structure was quite simple.

The software architecture

Two distinct aspects have been kept in mind in designing the software architecture: report generation and update.
For report generation, an extensive usage of the SYSTEM 2000 strings has been made. This approach gave sufficient flexibility, making easy the implementation of new reports and of simple help.

For update transactions, an ad hoc software was developed. Its architecture was based on the consideration that every application against the database could be seen as a navigation along the tree (SYSTEM 2000 is an hierarchical DBMS). This lead to the development of a general purpose, form based, software architecture. A general purpose I/O routine was accessing an application dictionary (SYSTEM 2000 based, too) in order to build a standard 3270 map. The application program had simply to ask for the appropriate schema record: the I/O routine was in charge of controlling every user/system interaction, and verify every constraint on the fields (list of values, uniqueness, range, etc.).
The application main program had simply to control the procedural constraints and the logic.
It is worthwhile to note that some constraints were implicit in the DBMS data model, and so automatically verified by the DBMS itself.
Some traditional programs were written for the batch updates.

The CASE tool approach

When the decision was taken to move from SYSTEM 2000 to SQL/DS, it was considered the opportunity of making a complete reenginering of the application. One of the most relevant decisions was to make use of an automated tool for the design and the documentation of the application.
The adopted tool has been IEW (Information Engineering Workbench).

IEW

IEW (Information Engineering Workbench) is one of the the CASE product leader in the market, and is one of the product selected by IBM in AD/Cycle.
It implements the approach of Information Engineering ([Martin87]), intended as everything that concerns information: information strategy planning, data model, subject area analysis, system design, etc..
The basic assumption is that analysis and design for computing can be tought as having a four level, pyramid structure, concerned with the two aspects of *data* and *processing*.

The *Information Strategy Planning* is concerned with the high level overview of an enterprise, its functions, and its information systems strategy. At this level, a strategic overview of data and functions is taken.
The *Business Area Analysis* is concerned with what fundamental processes are needed to run a business area, how they interrelate and what data are needed. It produces a fully normalized logical data model and a description of the necessary processes and their interdependence.
The *System Design* is concerned with how the processes are implemented as procedures, how the procedures work, and how they interrelate. The output of this phase is a design of the records used by procedures, and a design of procedures and how they relate.
The *Construction* is concerned with the design of programs and the implementation of data structures. The result are storage structure and program view of data, detailed program logic.
All these tasks are supported by a variety of diagramming techniques, that use, as far as possible, a consistent notation (Action diagrams, Entity-relationship diagrams, Decomposition diagrams, Data flow diagrams, etc.).
For each level of the DP pyramid, a specific workbench is available: they share the data stored in the *Encyclopedia*, which acts as a central repository of knowledge and allows the diagrammer tools to be integrated.

The usage of the tool

Even if a major emphasis has been put on the organization structure analysis, the size of the application was not so large to constitute a really effective test-bed. Anyway, the usage of the ***Planning Workstation*** produced one of the most relevant impacts of the adoption of the tool: a greater emphasis has been put on the aspects of analysis of the organization. As it has already been pointed out, this aspect was lacking in the manual methodology.

During the reengineering of the application, the opportunity of restructuring the conceptual schema was considered. This task was accomplished making use of the ***Analysis Workstation*****.** During this phase, the experiment of involving the programmers, in charge of developing the software, was conducted. Their contribution resulted very effective, as some of them were users of the previous application, and potential users of the new one.
The extensive use of the Analysis Workstation made evident the usefulness of an interactive Entity-Relationship editor, but pointed out some limitations. The Entity-Relationship model adopted by IEW does not show the semantic enrichements that we were used to have in the DATAID methodology.
In particular, the *generalization* concept is not supported: this fact forces the designer to take, at the conceptual design phase, some decisions that could be postponed up to the phase of logical design. The lacking of the IS-A hierarchies leads to two possible drawbacks:

- if the designer takes the decision of implementing an hierarchy as many entities, connected by optional relationships, the resulting conceptual schema appears to be graphically more complicated, as every relationship with the generalization entities must be splitted in as many relationships as the sub-entities are;
- if the designer takes the decision of implementing an hierarchy as a single entity, he must guarantee the semantic constraints that were implicit in the richer model. If we consider that there is no way, at this stage, to formally state the constraints, it can be easily seen that the risk of producing an inconsistent design exists.

On the other hand, the availability of *multiple attributes* makes possible to avoid the presence of useless entities in the schema. The designer may, anyway, make use of some special entities, named attributive entities. This feature is particularly useful in the aim of representing *aggregate attributes* (repetitive or non repetitive).

The *n-ary relationships* are not supported. The way out is to make use of a special kind of entities: the associative entities. These are used in order to represent relationship with attributes, too.

Procedural aspects are represented by means of the well known *data flow diagrams* and *mini-specs*.
It is known that Data Flow Diagrams acts as a "meeting point" for the *function-driven* approach and the *data-driven* approach, but are not suitable for the representation of the temporal sequence of events and procedures. To some extent, this task may be accomplished by an overloading of the concept of Data Store, but this practice should not be encouraged.

The use of the ***Design Workstation*** allowed the generation of the logical database schema (i.e. the DDL for the definition of the SQL/DS tables). This was an easy task.
What should be noted, viceversa, is that the mapping between the conceptual schema and the logical schema is fully automatic, and there is no way the user may affect this process (for example, put togheter two entities linked by a 1:1 relationship, on the basis of performance considerations like the physical storage size, the frequence of access to both entities, and so on).

As far as the implementation is concerned, it was decided not to adopt a conventional third generation language, but to switch to a fourth generation language: NOMAD2 was the choise.
As a consequence, no use has been made of the ***Construction Workstation***. This tool, anyway, is generating code for the DB2-CICS-COBOL environment.

As far as the ***Documentation*** is concerned, it should be noted that, while the graphical representation of data is very effective, the textual reports appear to be verbose. There is no use of

special styles or character size, and sometimes may be difficult to find the right information in a so large amount of paper.

Even if the vendor assert that IEW is a fully integrated tool, we had to notice that there are few cases (i.e. data types) where the vertical integration between the workstations appears to be scarce, as the modifications made at lower levels are not automatically exported toward higher levels. This fact is particularly tedious because, in the MS-DOS version, it is necessary to quit one workstation before activating another one. May be this aspect will be greatly affected by the migration in OS/2 environment.

Evaluation of the experience

From the comparison of the two approaches, we may stress the following aspects:

- *Analysis of the organization.*
  This aspect was not considered in the manual methodology. The use of the tools supplied by the Planning Workstation gave a valuable help in considering some aspects that were not so much emphasized in the first implementation of the application.

- *Data Analysis and constraints representation.*
  The DATAID methodology is adopting a semantically richer data model, which is representing many implicit constraints. All other types of constraints, however, are simply annotated in free text form.
  As far as IEW is concerned, its excellent user interface makes possible to increase the participation of the users in the phase on conceptual schema design, even if the lacking of some aspects peculiar to the semantic data models makes some steps more difficult, and the resulting conceptual schema less readable.

- *Dynamic aspects*
  The DATAID methodology is representing dynamic aspects by means of a theorically solid formalism, that is, Petri nets. It must be noted, however, that the methodology is concerned more with data than with operations, and the filling of the forms and the glossaries is not an easy task. Moreover, their usage is suggested at a too low level of atomicity.
  IEW makes use of Decomposition Diagrams, Data Flows e Action Diagrams. These formalisms are widely used even by other Software Engineering tools, but are unsuitable to express temporal dependencies and alternatives.

- *Mapping.*

DATAID performs this task in several steps, going back to restructure the conceptual schema, on the basis of some general rules, plus some qualitative choises made by the database designer. IEW automatically produces the database structure and the DDL, but the user has the possibility of modifying the mapping rules only acting on the conceptual schema.

- *Development time*.
  For a project of this size, the development time has not been considerably reduced.

- *Quality*.
  Estonishingly, the overall quality of the application does not seem to have been considerably improved, in spite of the migration from a hierarchical to a relational DBMS, and the adoption of a fourth generation language. After all, a careful database design and a good software architecture are the key factors for the success of an application.

As a conclusion, we have however to notice that the the adoption of a CASE tool leads to a general improvement of the understanding of the application, a greater accuracy of the data structures design, a clear software architecture, a detailed documentation. It is obvious that these improvements are more relevant if the automated tool is adopted for projects of significative size (even in terms of the size of the development group) and also is the mean for introducing the usage of a more formal design methodology.

References

[Chen76] Chen P.P.: *The Entity-Relationship Model: Toward a Unified View of Data*, ACN TODS, Vol. 1, N. 1, (1976), pp. 9-36

[Ceri83] Ceri, S. (ed.): *Methodology and tools for data base design*, North-Holland (1983)

[Martin87] Martin J.: *Recommended Diagramming Standards for Analysts and Programmers: A Basis for Automation*, Prentice-Hall (1987), ISBN 0-13-767377-9