A methodology for the design of data dictionaries

C. Batini, G. Di Battista, G. Santucci

Dipartimento di Informatica e Sistemistica Universita' di Roma "La Sapienza"
Via Buonarroti 12 - 00185 Roma, Italy

Work partially supported by Progetto Finalizzato Calcolo Parallelo e Sistemi Informatici of CNR and Arthur Andersen Co.

ABSTRACT

More and more organizations perceive the importance of having a better comprehension of data, seen as a strategic resource. The data dictionary of the organization is the integrated repository of all types of data produced, managed, exchanged and maintained in the organization, represented in such a way to allow all users of the information system to understand their meaning. Since the way data are represented is usually dependent on the application domain, the DBMS used, the role of the responsible organization unit, designing and maintaining a dictionary is a challenging task. In the paper a methodology is discussed for data dictionary design; several structuring mechanisms are proposed, to be used in the representation of data in the dictionary. A strategy is provided to produce the dictionary, according to such mechanisms.

1. Introduction

It is thoroughly admitted that data are a strategic resource of organizations, since they are more stable in time than processes, and the organization itself. At the same time, modern information systems tend to be more and more complex, and are characterized by several heterogeneous aspects, such as:

1. different DBMS models used to represent data, such as the hierarchical, network, relational models; extending the argument, spreadsheets, multimedia databases, knowledge bases are other types of models currently used in information systems, each of them providing its own representation of data;

2. different abstraction levels adopted by users in representing data of interest. Top management sees human resources in terms of managers and employees, while the personnel office sees them classified in terms of top managers, middle managers, secretaries, analysts programmers, etc;

3. different levels of aggregation; users, according to their position in the organization "pyramid" look at data in terms of different levels of summarization, from elementary data, to statistical indexes. E.g. the budget is seen by top management in terms of estimated and final balance, by administration in terms of detailed input-output items;

4. different roles of organization units involved in data management; usually, the units responsible of data design, creation, manipulation, access, update are distinct.

As a consequence of the above heterogeneous aspects, it is a challenging issue to allow users of the information system to understand the meaning of all types of data that circulate in the organization. This is made possible by the availability of a data dictionary. By data dictionary we mean the integrated repository of all types of data produced, managed, exchanged and maintained in the organization, represented in such a way to allow all users to understand the meaning of the data resource. It is also part of the data dictionary the representation of relationships among data and the other resources involved in the information system, i.e. processes, organization units, technologies.

Creating and managing a data dictionary is becoming of growing importance in complex organizations. Present so called "data dictionaries" allow usually representing only metadata that concern application aspects of the information systems, namely, files, procedures, etc. with low concern to the conceptual characteristics of data, i.e. aspects related to the meaning, independently from their implementation and usage in processes and organization units.

In this paper we propose a methodology whose goal is to establish a precise and clean strategy for creating and maintaining data dictionaries. Two are the main aspects involved in the methodology: the structure of the dictionary (section 2), and the strategy to build it (section 3). The paper has to be seen as a first contribution in a complex research area; previous related topics appear in [2] and [13]). Concluding remarks will point out open research problems in the area.

2. The structure of the data dictionary

2.1 Resources, models, schemas

The data dictionary has the goal of representing data, and their relationships with other relevant resources of the organization. This means that data should be seen by users both as an independent resource and in their interaction with processes, organization units, technologies.

Each resource of an information system can be represented by a suitable set of models, each one emphasizing specific characteristics and properties. It is our opinion that, also inside a given resource, no universal model exists that can represent in a uniform way all meaningful properties. For example (see fig 2.1):

1. referring to data, the Entity Relationship (E-R) model is suitable for elementary data, that describe the primary activities of the organization; for aggregate or statistical data the E-R model reveals ineffective, and we need a model where aggregations can be expressed.

2. for processes, we can use Data Flow Diagrams [16], [11] to represent relationships among processes and information flows, Petri Nets [5] to represent causal relationships among processes.

3. concerning organization, we can use organization charts to represent the hierarchical or functional structure, Pert diagrams

CH2799-5/90/0000/0706$01.00 © 1990 IEEE

to represent the temporal allocation of resources in a project.

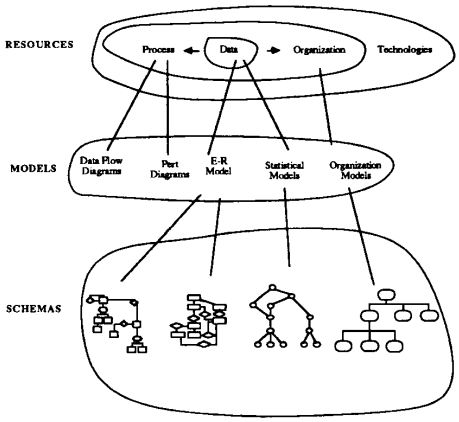

Fig 2.1: Resources, models, and schemas in the Information System environment

If we accept the above position, the important point in building a dictionary is being able to find a minimal set of models, and establish precise rules to migrate between such models. This problem is not new in the literature on information systems, while a mature solution is far to be achieved. To give some examples: strategic planning methodologies suggest the use of matrices relating data, processes, organization; several methodologies for data and functions analysis (see [1], [16], [11]), with different formalisms establish a bridge between data and functions; in [2] a formal interface is described between schemas that represent data at the elementary and at the aggregate level.

Every model is used in the dictionary to represent a certain number of schemas, high level descriptions of a resource of interest. Schemas are "pictures" of a piece of the information system, in the framework of one specific resource, and one specific model. Together with schemas, in the dictionary we want to represent relationships between them. Such relationships can be classified as intraresource vs interresource, or intramodel vs intermodel, likewise. The goal of the following section is to describe in detail the types of relationships suitable for structuring a data dictionary.

2.2 Structuring primitives

The structure of a data dictionary subsumed by the above arguments is certainly complex; so, we need to define structuring primitives that allow to manage the overall set of schemas that populate the dictionary. Four structuring primitives are proposed, that can be considered as a minimal set needed to capture the relationships mentioned in paragraph 2.1. They are: refinements, views, derivations, integrations. They are not to be seen as an exhaustive list, rather the most important among an open ended set of mechanisms. We discuss them separately.

Refinements

The refinement is the mechanism that allows to model the same piece of reality in terms of several descriptions at different levels of detail. Refinement is the typical paradigm used in so called top-down methodologies developed for functional analysis. More recently, similar methodologies have been developed for data analysis, thus establishing a formal framework for top-down data refinements. In [2] a set of data refinement primitives is defined, chosen for their simplicity and context free behavior as the most suitable generating mechanisms of schemas at different levels of detail. We recall them informally by means of an example (see fig 2.2), that refers to the customers office of a bank. For every pair of schemas Si and Si+1, refined version of Si, we show in Si+1 by means of closed lines the result of refinements performed on concepts in Si: corresponding primitives are easily determined.

Refinements, as introduced above, are intramodel and (as a consequence) intraresource relationships.

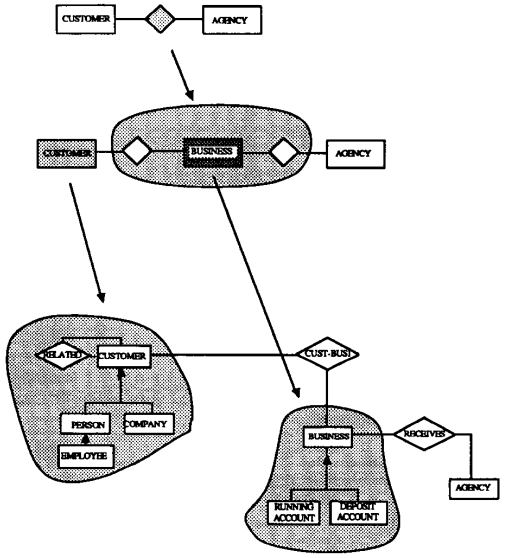

Fig 2.2 Examples of refinements and refinement primitives for a banking customer schema

Views

Views are customized versions of fragments of a schema. Views can be used, for example, to relate the schema of the credits office, and the subschema referring to customers, businesses and agencies (see fig 2.3). Views are usually built by means of a formal query language defined on the model. They are intramodel and intraresource relationships.

Integration

Integrated schemas conveying the information content of two or more schemas in the dictionary are needed, in order to get a global view of data managed by a whole information system, an organization area, a distributed system. Methodologies for schema integration have been discussed in detail in the literature (a comparative review of such methodologies appears in [3]); we will see examples of integration in the framework of a data dictionary in section 4. Integration is an intramodel and intraresource relationship.

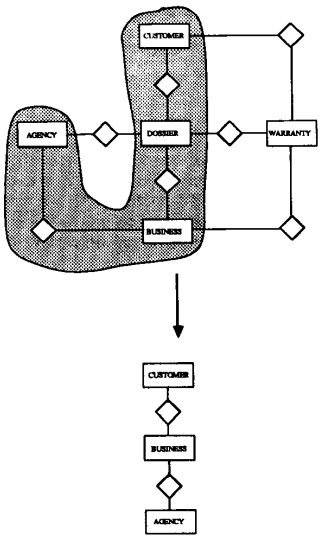

Fig 2.3 An example of view

Derivation

Derivation is the mechanism needed in order to move from one model to another in representing a certain piece of reality. More formally, given a pair of schemas S1 and S2, S1 expressed in terms of a model M1, S2 in terms of a model M2, a derivation maps a concept of S1 into a view of S2. For example, we may put in correspondence:

1. a data flow in a Data Flow Diagram, and the corresponding set of concepts in a E-R diagram;
2. an entity in an E-R diagram with the corresponding relational view in a DBMS schema (see fig 2.4).

Derivation is an intermodel structure, and can be an intra (case 2 above) or interresource (case 1) structure, according to the types of models involved.

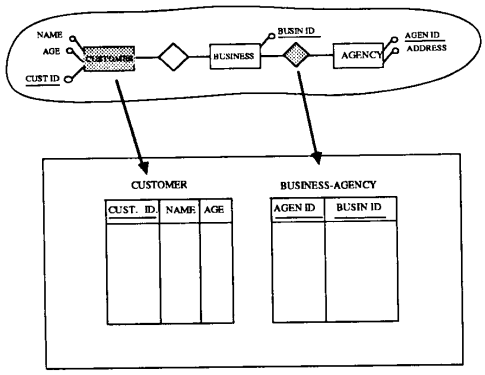

Fig 2.4: An example of derivation

In order to summarize the above logical relationships, we show in fig 2.5 a taxonomy of structuring primitives in terms of:

1. resource vs model,
2. intra vs inter types.

RELATIONSHIPS	MODEL		RESOURCE	
	intra	inter	intra	inter
refinement planes	X		X	
views	X		X	
derivations		X	X	X
integrations	X		X	

Fig 2.5: Taxonomy of structuring primitives

2.3 Structure of the organization and structure of the dictionary

The data dictionary represents the data resource managed in the organization in an integrated fashion. From one side, data concepts are to be represented as a global resource, independently from processes, organization units, implementations; from the other side, we need to represent the way in which data are managed "here and now". So, the structure of the dictionary can be seen as made by three layers (see figure 2.6):

1. a global layer (top of the pyramid), where data and properties of data common to the whole organization are represented. In this layer, data are described at a very abstract level, in order to get a global, uniform representation (one or more global skeleton schemas at different refinement levels). It is worth noting that in the attempt to choose concepts meaningful for this layer, it may happen that only very abstract (and as a consequence, not meaningful) concepts are found; in this case, the global layer is better represented in terms of a small set of loosely coupled skeleton schemas. At this layer, the most common structures to be used are refinements and integrations.

2. a local layer (bottom of the pyramid) where data are represented as they are seen in specific information systems, and specific organization units. At this layer, all types of structures are meaningful.

3. an intermediate layer, whose goal is to allow a graduated ford from the integrated representations of the global layer and the tailored representations of the local level. Together with refinements, views are the typical mechanism to be used at this level, in order to relate schemas referring to different information systems and organization units.
The above three levels architecture shows suitable in small/medium organizations; in complex organizations, where areas exist with high cultural cohesion, and a highly independent organization structure, it is worth to re-propose the above architecture for the local layer of such areas (see fig 2.7).

3. A methodology for the design of data dictionaries

We address now the problem of building a data dictionary based on the above assumptions. The structure of the methodology is shown in fig 3. In the most general situation, applications presently managed in the organization may be documented by means of:

- a logical schema, usually represented in the data dictionary of the DBMS.

- a conceptual schema, usually represented with a CASE tool.
- a full set of refinement planes, produced during top down design.

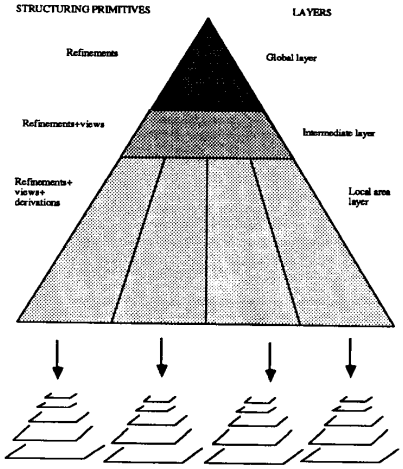

Fig 2.6: Global structure of the data dictionary

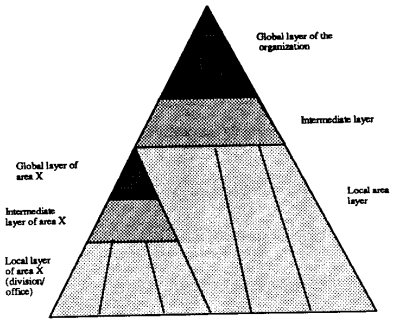

Fig 2.7: Recursive structure of the dictionary

The first case includes the two main activities of "preprocessing", namely reverse engineering and documentation.

The goal of the reverse engineering step is to reconstruct the conceptual schema of the application. from the logical schema. Several methodologies have been proposed in the literature for different situations: the mapping can be hierarchical to classical E-R (i.e. the E-R model defined in [8]) [14], relational to classical E-R [6], flat files to classical E-R [19], relational or hierarchical to E-R enriched with generalization hierarchies [20].

The goal of documentation is to produce a set of refinement planes starting form a conceptual schema. A methodology for such activity is shown in [2], where a taxonomy of primitives is proposed for refinement activities, and several criteria for checking the quality of refinement planes are discussed.

In the three cases a filtering step is also needed whose goal is filtering from schemas all "technical" concepts that are meaningful for application purposes and aren't of interest in the dictionary.

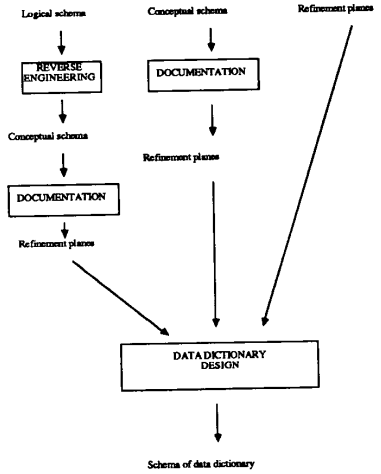

Fig 3.1 General structure of the methodology

Reverse engineering, documentation, filtering are activities that in a sense should precede the dictionary design activity. We will not deal further with them in the following, referring to the literature for further details.

DICTIONARY DESIGN (RP1, RP2, .., RPn, DICTIONARY)

Input: A set of sets of refinement planes RP1, .., RPn, , each one related to a specific application managed in the organization.
Each RPi is composed of schemas Si_1, .., Si_{mi}, (i=1,..,n) which describe the application at different refinement levels.
Output: The DICTIONARY, expressed in terms of a structured set of:
1. refinements, 2. views, 3. integration planes, 4. derivations, according to the general architecture described in section 2.

(* Produce refinement levels for the global schema *)
1. CHOOSE a suitable intermediate cut of planes RP1, ..., RPn, producing a set of schemas $S1_{k1}$, .., Sn_{kn}, candidates for integration.
2. INTEGRATE $S1_{k1}$, .., Sn_{kn} into Gk, intermediate refinement of the global schema.
3. PROPAGATE Gk toward the top and the bottom of the hierarchy of refinements, using new local schemas, chosen over and under the cut $S1_{k1}$, .., Sn_{kn}.
Stop the production of schemas more abstract than Gk when concepts needed are proportionally too abstract, and, as a consequence, not meaningful.
Stop the production of schemas more refined than Gk when the size of the schema become unfeasible.
Call G1, G2, ..., Gmg the new hierarchy of schemas, choosing mg > max (m1, m2, .., mn).
Call CH the set of schemas used in RP1, ..., RPn to build such hierarchy.

(* Produce the refinements/views of local schemas *)
4. PLACE each schema in CH in the refinement/view grid (such schemas, by construction, are both a refinement of some schema of an application, and a view of some global schema, at a certain refinement level).
5. PLACE schemas not in CH in the refinement/view grid, in case moving up or down the refinement level of the schema.

(* Produce derivations *)
6. BUILD derivations for each final refinement.

Fig 3.2 A strategy for the design step

The structure of the methodology for dictionary design is shown in fig 3.2.

In principle, the design strategy can be simply described as follows. First produce an integrated representation of the global schema (or in case, of a limited set of schemas, corresponding to partial integrations); then produce the whole set of refinements for the global schema; now, review and in case restructure local schemas to place them in a hierarchy of refinements and views. Finally produce derivations, at different refinement levels.

In describing the methodology, we use an example, based on a banking application. We assume that the bank is organized in terms of a set of divisions. We are interested in the "loans" division, that can be seen as composed of three functional areas, (see in fig 3.3 the corresponding organization chart), namely "customers", "credits", and "warranties".

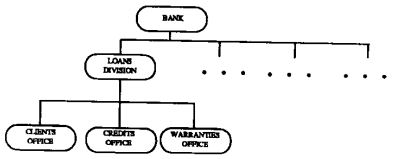

Fig 3.3 Organization chart for the areas of the case study

The customers office deals with personal informations on customers; the credits office deals with proposed credits, and corresponding dossiers prepared in order to take a decision; the warranties office deals with all types of warranties that can be used by costumers to get a credit. The three corresponding data schemas are shown in fig 3.4.

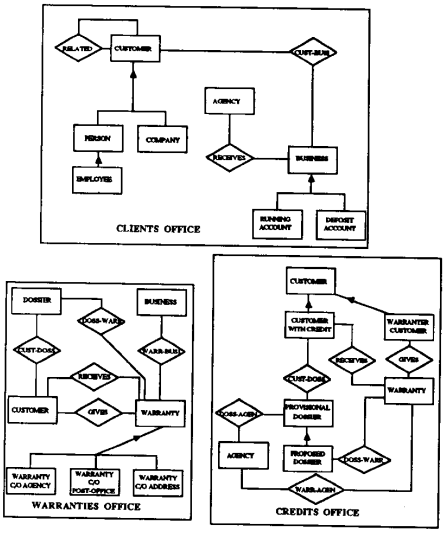

Fig 3.4. Case study: three schemas for a banking information system

Before addressing specific steps of the methodology, we need to perform a preprocessing activity (see again fig 3.1), whose goal, in this case, is to produce refinement planes. Concerning the customer schema, we have already shown in fig 2.2 the structure of refinements. In fig 3.5, 3.6 we show refinements for credits and warranties. Notice that the credits schema is more complex than others, having four refinements plus the final schema.

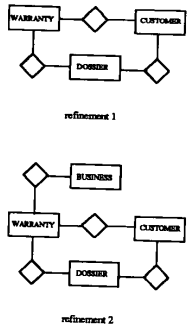

Fig 3.5: Refinements for the warranties schema

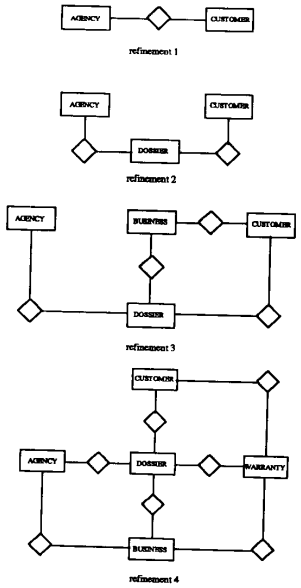

Fig 3.6: Refinements for the credits schema

3.1. Produce refinement levels for the global schema

At this point, as suggested by step 1 of the methodology, we have to choose a cut of refinement planes that looks suitable for integration. The idea here is that, due to different sizes of the information systems operating in the bank, and the different roles in the organization, we have to choose the abstraction level at which concepts are more easily mergeable. In our

example we can choose (see fig 3.7) planes customers2, warranties2, and credits3, that give rise to a schema similar in size to each of the three previous schemas. A bad choice corresponds to merge customer1, credits1, warranties1, since the corresponding integrated schema looks too unbalanced toward the warranties schema; in other words, warranties1 is at a refinement level far more detailed than customer1 and credits1, and no schema more abstract than warranties1 can be found for which integration with customer1 and credits1 is feasible.

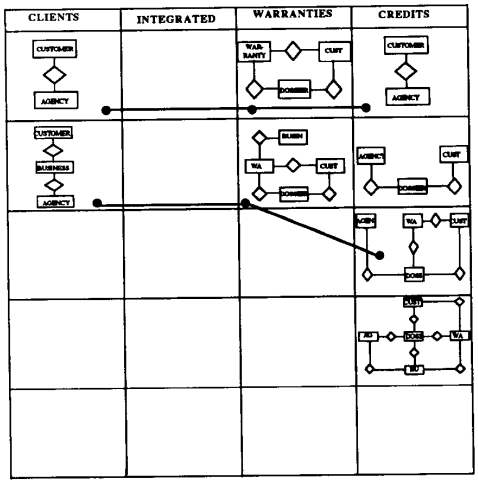

Fig 3.7: Possible choices of cuts in refinement planes

Once chosen a first cut, we have to integrate schemas (step 2). Methodologies for schema integration have been deeply studied in the past. In [4] a methodology for integration of E-R Model schemas is described. In general, it may happen that local schemas can be incoherent each other: synonyms, homonyms, different types for the same concepts are to be checked and resolved.

Now we have to build the remaining refinements for the integrated schema. A rough idea on the number of planes more and less refined than the first schema is given by the population of refinements over and under the chosen cut. In producing new global refinements, we have to choose new local schemas, that may in case coincide with schemas chosen previously.

The production of refinement planes for the global schema can become unfeasible for large organizations. When the global schema tends to be too large, we can suspend the generation of global refinement planes, and proceed recursively to produce local dictionaries, as suggested in fig 2.7. On the other side, it may happen that high level refinement planes use very abstract, and, as a consequence, not meaningful concepts. In this case, it makes no sense to force the production of such levels, and it is better to "cut the pyramid", keeping not integrated the high level local schemas. We do not stress the example, being of small size, to keep into account these situations. In fig 3.8 we show refinement levels for the integrated schema. The final integrated schema appears in fig 3.9.

3.2. Produce refinements/views of local schemas

The goal of this step is to consolidate each local schema in the grid of refinements/views. Each column of the grid (see fig 3.10) represents an application or functional unit of the

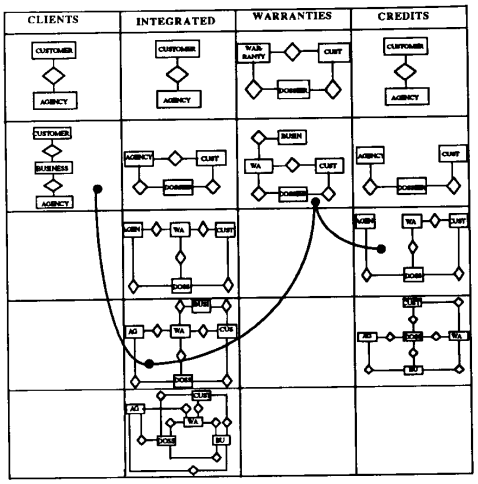

Fig 3.8: Intermediate refinements for the integrated schema

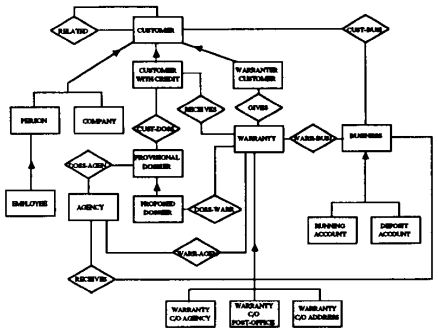

Fig 3.9: Final refinement for the integrated schema

organization, while each row represents an abstraction level. Schemas are placed in the grid in such a way to put in the same row schemas at the same abstraction level. Clearly, the above schemas (local schemas chosen for integration + the integrated schema) are also placed in the same row. Also final schemas are placed on the same (last) row. Due to the definition of view, a schema S1 that was integrated into a global schema GI, is dually a view of GI.

In placing schemas according to the above rules, it may happen that some local refinement is never chosen, and so is placed nowhere; in this case we have to suitably modify the schema, abstracting, refining or equivalently transforming some concept in order to obtain a new schema that can be placed in the grid. The final grid is shown in fig 3.11.

3.3 Produce derivations

We are now ready to produce derivations. Here we show the case of derivations for statistical data. We assume that statistical data are represented in terms of the Conceptual Statistical Model, described in [12]. We assume that schemas are populated with attributes such as at least:

1. Type, Sex, and Class of Activity for Customer;
2. Name for Agency.

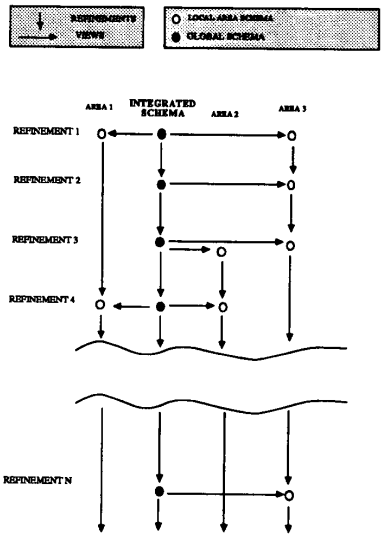

Fig 3.10: Relationships between refinements and views

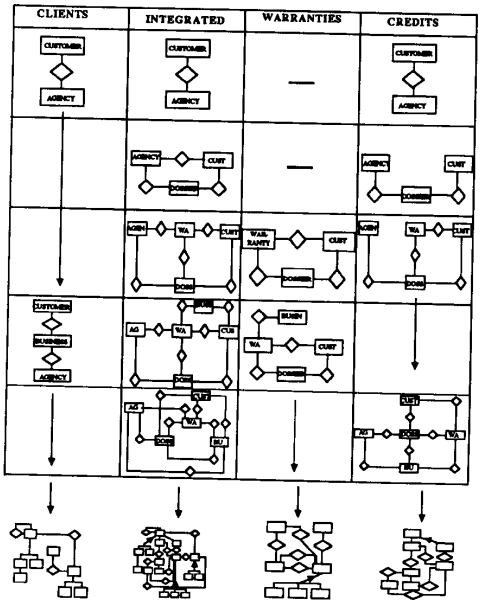

Fig 3.11: Refinements/views grid for the example

Assume we want to show statistical aggregations defined on Customers, like:

a. Average Age of Customers, classified by Type, Sex, and Name of Agency;

b. Number of Customers, classified by Type and Class of Activity.

The corresponding derivation is show, in terms of the diagrammatic representation of CSM, in fig 3.12

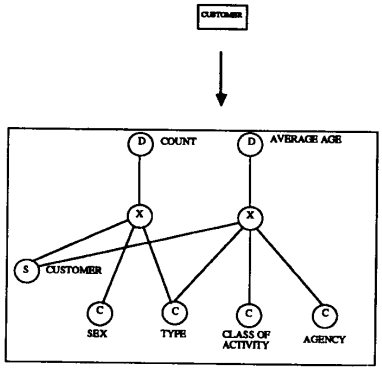

Fig 3.12 Derivation of a statistical schema from the global data schema

The above mechanism can be uniformly applied to all other types of models represented in the dictionary, like DFDs, Petri nets, and so on.

4. Future developements

We have presented a proposal for the structure of a data dictionary, and a methodology for its initial design. Some open problems still remain, concerning:

1. the impact of the dictionary in the organization. A new role of dictionary administrator has to be suitably defined, both centrally and in specific organization units, whose goal is to interact both with the user and with the analyst for dictionary design and maintenance.
2. In the paper we have addressed the data dictionary as a passive tool, useful to understand how data are managed in the organization. The dictionary should have also an active role, especially in information system planning activities. While several CASE tools suggest using high level data representations in strategic planning [18], mature methodologies are needed for this purpose.

Another difficult issue concerns the management of a huge amount of diagrams in the dictionary. In order to allow the user navigating among the whole set of schemas, using the structuring mechanisms discussed in the paper, it is necessary to produce dynamically diagrams starting from internal descriptions of their topological structure. Several papers have been written in the last years on automatic graph drawing (see [21], [22]), and some CASE tools provide functionalities that allow automatic layout of diagrams (see [17]).
The navigation in the dictionary is made easier having at disposal user friendly languages for expressing the navigation path. Promising results are provided by "Query by Diagram" languages, (see [15], [7]), where E-R diagrams are the interface mechanism at disposal for the user to express queries.

REFERENCES

[1] C. Batini, S. Ceri - Joint Conceptual Design of data and

functions - in (R. Wagner, R. Trausmuller, H. Mayr eds.) Informationsbedarsf ermittlung und analyse fur den Entwurf von Informationssystemen, Springer Verlag, 1988.
[2] C. Batini, G. Di Battista - A Methodology for Conceptual Documentation and Maintenance - Information Systems, Vol 13, N. 3, pp. 297-318, 1988.
[3] C. Batini, M. Lenzerini, S. Navathe - A Comparative Analysis of Methodologies for database Schema Integration - ACM Computing Surveys, September 1986.
[4] C. Batini, M. Lenzerini - A methodology for data schema integration in the Entity Relationship model - IEEE Transactions on Software Engineering, November 1984.
[5] W. Brauer, W. Reisig , G. Rozenberg (eds.) - Petri Nets: applications and Relationships to other models of concurrence - Lecture Notes in Computer Science 255, 1987.
[6] M. Casanova, J. Amarel de Sa - Designing Entity Relationship Schemas for Conventional Information systems - - in (P. Chen ed.) Entity Relationship Approach to Software Engineering, North Holland, pp 265-278, 1983.
[7] T. Catarci, G. Santucci - Query by Diagram: a Graphic Query System - in (C. Batini ed.) Proc. Seventh International Conference on the Entity Relationship Approach, North Holland, 1989.
[8] P.P.S. Chen - The entity relationship model: toward a unified view of data - ACM Transactions on Data Base Systems, Vol. 1, N. 1, 1976.
[9] K.H Davis, A.K. Arora - A methodology for translating a conventional file system into an entity relationship model - Proc. Fourth International Conference on the Entity Relationship Approach, Chicago, 1985.
[10] K.H. Davis, A.K. Arora - Converting a relational database model to an Entity Relationship Model - in (S. March ed.) Proc. Sixth International Conference on the Entity Relationship Approach, North Holland, 1988.
[11] T. De Marco - Structured Analysis and System Specification - Yourdon, 1979.
[12] G. Di Battista, C. Batini - Design of Statistical Databases: a Methodology for the Conceptual Step - Information Systems, Vol 13, N. 4, pp. 407-422, 1988.
[13] G. Di Battista, H. Kangassalo, R. Tamassia - Definition Libraries for Conceptual Modelling - in (C. Batini ed.) Proc. Seventh International Conference on the Entity Relationship Approach, North Holland, 1989.
[14] S. Dumpala, S. Arora - Schema Translation using the Entity Relationship approach - in (P. Chen ed.) Entity Relationship Approach to Information Systems Modeling and Analysis, North Holland, pp. 337-356, 1983.
[15] R. El Masri, J. Larson - A Graphical Query facility for ER Databases - Proc. Fourth International Conference on the Entity Relationship Approach, Chicago, pp 236-245, 1985.
[16] C. Gane, T. Sarson - Structured Systems Analysis - Prentice Hall, 1979.
[17] GESI Srl. - Master User Manual, Roma, 1988
[18] J. Martin, E.A. Hershey 1986 - Information Engineering: a management white paper - Knoledgeware Inc, 1986
[19] E.G. Nillson - The translation of a Cobol data structure to an Entity Relationship type conceptual schema - Proc. Fourth International Conference on the Entity Relationship Approach, Chicago, 1985.
[20] S.B. Navathe, A. Awong - Abstracting Relational and Hierarchical Data with a semantic data model - in (S. March ed.) Proc. Sixth International Conference on the Entity Relationship Approach, North Holland, 1988.
[21] P. Eades, R. Tamassia - Algorithms for Drawing Graphs: an Annotated Bibliography - Submitted for publication, 1989.
[22] R. Tamassia, C. Batini, G. Di Battista - Automatic Graph Drawing and Readability of Diagrams - IEEE Transactions on System, Man and Cybernetics, January 1988.