ソフトウェア工学 1−2
（1977. 5. 6.）

Database System Design Criteria
1. Specification of Basic Design Requirements for Application Independence.

Tosiyasu L. Kunii

Department of Information Science
The University of Tokyo
Hongo, Tokyo 113 Japan

ABSTRACT

This is the first of a series of papers discussing database system design criteria. Application independence, data independence and machine independence as the basic design criteria are derived from database system design requirements. In this paper applications are considered rather in detail as a provision for establishing application independent design criteria more precisely.

1. DESIGN OBJECTIVE

The design objective here is to provide a set of engineering tools for designing database systems (DSs). Engineering tools are as realization of generalized design methodologies for the most general cases, not for individual systems.

Then, design for individual systems are instances either of results of design tool applications, and/or special case generations by employing machanized generators with mechanized optimizers.

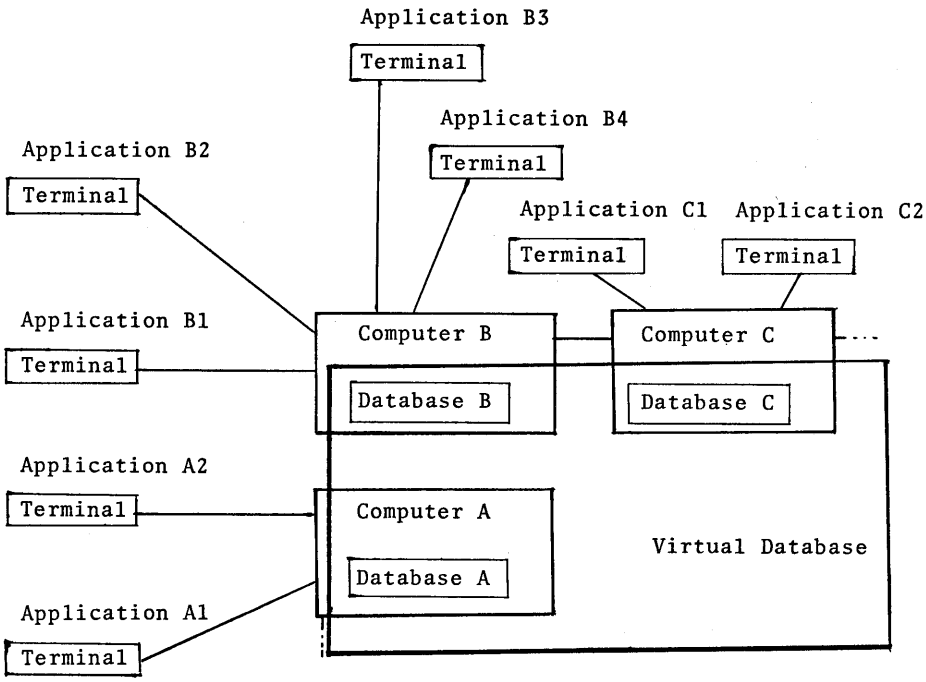


Fig. 1. A DS as a Virtual Information System

2. SPECIFICATION OF GANERAL DS STRUCTURE

General DS structures can be abstracted schematically as shown in Fig. 1. The scheme summarizes the most general situations where diversified applications are handled on heterogeneous machines linked togehter to utilize very large amount of data.[1] Here, data, applications, and machines are going in and out of a DS freely. Hence, a DS consists of data, applications, machines and a controller called a database management system (DMS, or DBMS). Looking this from a different angle, we can say that DBMS environment in a DS is as shown in Fig. 2.

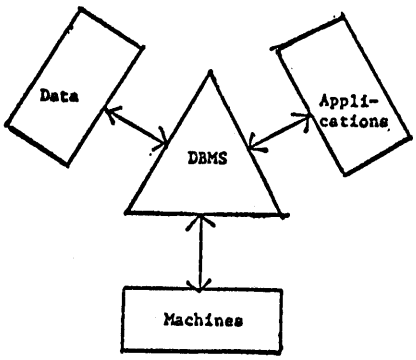


Figure 2 DBMS Environment

3. DS DESIGN CRITERIA

It does not pay to build different DBMSs individually for different kind of applications, data and machines. Furthermore, to do that is practically meaningless in the environment specified before, because then the tremendous number of different DBMSs have to be maintained within a DS or always have to be swapped in and out of a DS. Only practical way to design DBMS is to do it so that DBMS architecture is unaffected by the change of applications, data and machines which the DBMS has to control. In other words, DBMS requirements from the environment give the following DS design criteria:

1. Application independence,
2. data independence,
and
3. machine independence.

The following is a short list of what can be achieved by meeting the criteria.

Application independence:

1. New applications can be added without system architecture change,
and
2. adding new applications causes minimum system overhead.

Machine independence:

1. Easy replacement of machine,
2. heterogeneous network oriented architecture,
and
3. portability.

Data independence:

1. Free addition, deletion and update of data values,
2. free accomodation of new data structures and their instances,

and

3. free accomodations of new data control rules.

Difficulties of achieving the forementioned independences come from the fact that:

> Application models, data models and machine models are always changing to certain extent.

4. SPECIFICATION OF APPLICATIONS AS DBMS ENVIRONMENT

We have to specify DBMS environment in more detail so that we can derive more detailed DS design criteria. In this paper, only applications are discussed as the environment. The rest of the environment is considered elsewhere.[1]

Applications are a set of users' requirement specifications. The requirements here are those to data and machines. Requirement specification techniques thus far developed use either hierarchical graph formalism as seen for example in SADT of SofTech (see Fig. 3) [2], or bipartite graph formalism as seen for example in RSL of TRW [3], combined with hierarchical modular design techniques.[4]

MODEL BUILDING VIA STRUCTURED DECOMPOSITION

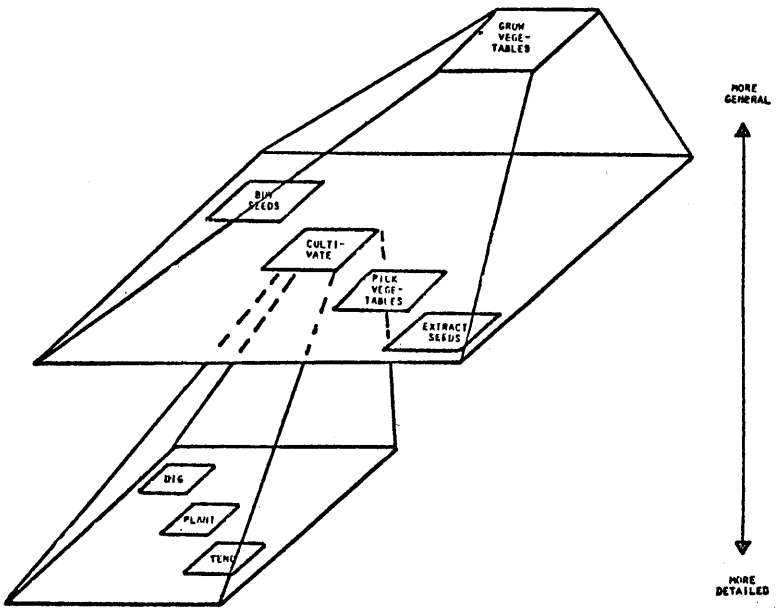


Fig.3 SADT (Structured Analysis and Design Technique) Example

Query languages for information retrieval (IR) also have certain similar capabilities. For example, SEQUEL of IBM San Joe Research [5] has hierarchical and recursive definition capabilities.

The design to satisfy the criterion of application independence can be realized, by having generator within a DS to automatically generate and update application dependent DS-interface. A commercial relational DBMS, ADABAS, of Software AG has actually realized application independence by maintaining the system generated application interface called the ASSOCIATOR which consists of different couplings and indexes of the relations for different applications.[6] This ASSOCIATOR is equivalent to what can be specified by LSL (Link and Selector Language).[7]

Another application independent design is to have a system generated virtualizer in a DS which maps applications to virtual resources and virtual processes hierarchically until the mapping hits on real processes and resources (Fig. 4). An example of the MIT CONIT system is given in Fig. 5. The ADABAS architecture has also a feature of virtual design. Each application, specified by relational schemata and queries, is mapped by a virtualizer called ADAMINT first to the virtual relations as specified by the schemata and the operations on them, then down to coupled relations and their indexes, until finally to real resources and operations which consist of an ASSOCIATOR generator, relational database and relational operations.[6]

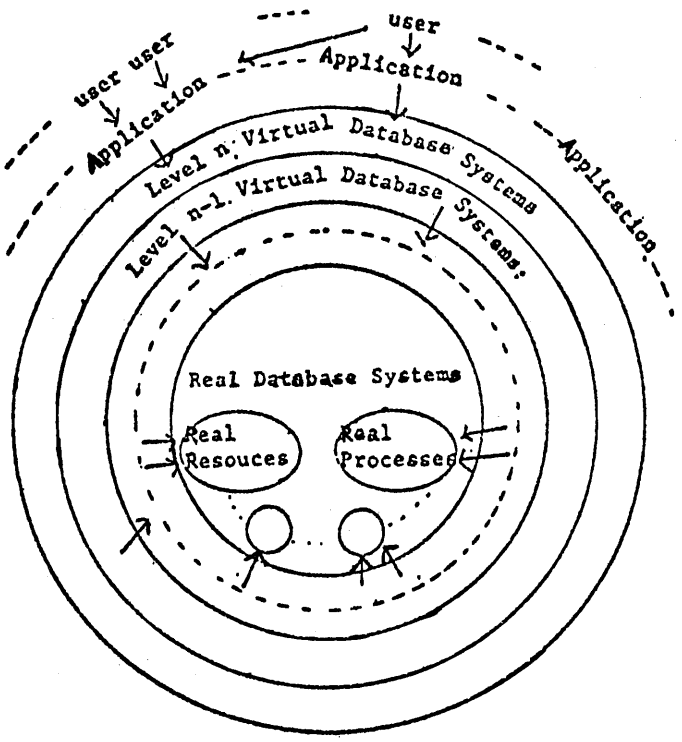
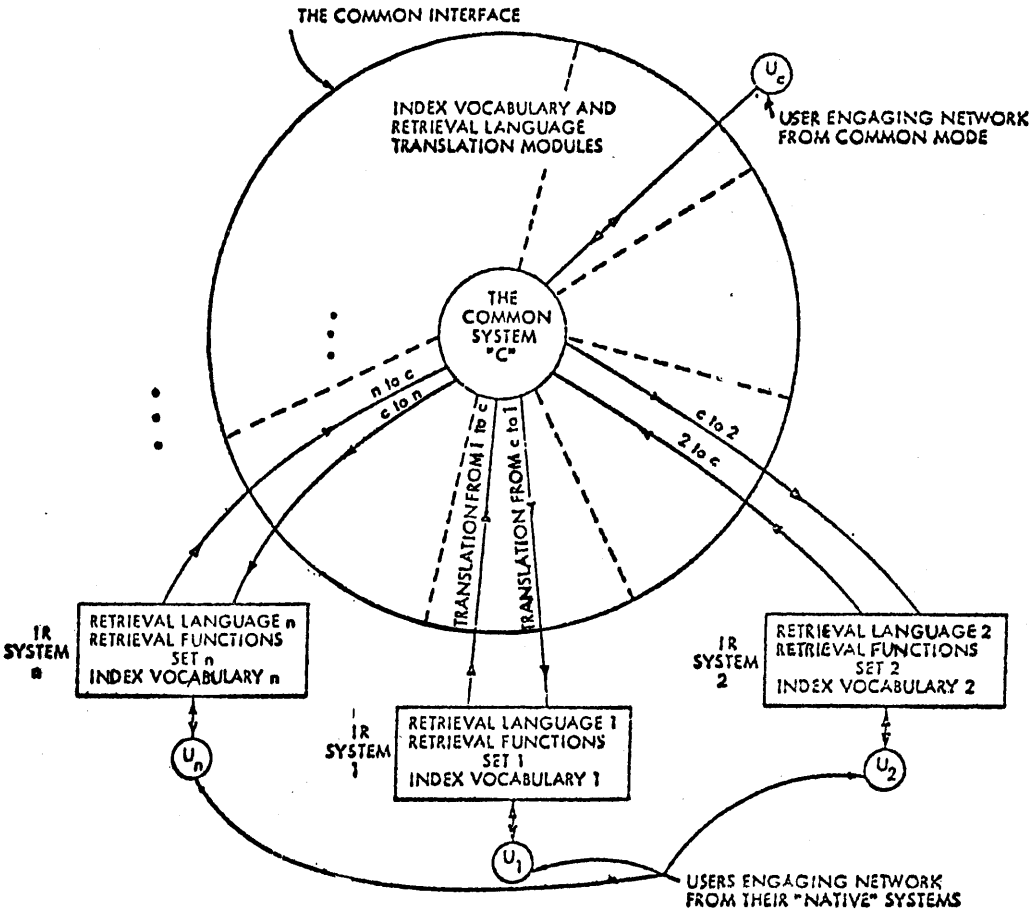


Fig.4. Virtual Design for a Multi Application Oriented DS



This figure describes the logical relations of an IR network based on a common interface for heterogeneous systems

Fig. 5

MIT CONIT (Connector for Network Information Tranfer) on MULTICS as an Example of Virtual Design
(a) CONIT structure

CONIT	DIALOG	INTREX	ORBIT
PICK SYSTEM (1) PICK DATA FILE SHOW INDEX TERM	[LOGOUT/LOGIN] (2) .FILE FILE NO. (3) EXPAND TERM	[LOGOUT/LOGIN] (2) NA (4) NA	[LOGOUT/LOGIN] (2) "FILE FILE NAME" (3) "NEIGHBOR TERM"
FIND TOPIC FIND AUTHOR NAME FIND A AND B	SELECT TOPIC SELECT AU=NAME SELECT A (C) B	SUBJECT TOPIC AUTHOR NAME SUBJECT A AND B	"FIND TOPIC" "FIND NAME (AU)" "FIND A AND B"
COMBINE SET 2 OR SET 3 SHOW SHOW TITLE	COMBINE 2 + 3 TYPE x/2 (5) TYPE x/6 (5)	S2 OR S3 OUTPUT OUTPUT TITLE	2 OR 3 "PRINT" "PRINT TI"
SHOW OFFLINE SHOW SET 3 SHOW DOCS 3 - 6 SHOW NEWS	PRINT x/2 (5) TYPE 3/2 TYPE x/2/3-6 (5) ? NEWS	OUTPUT OFFLINE S3/OUTPUT DOCS 3-6/OUTPUT NA	"PRINT OFF-LINE" "PRINT SS 3" "PRINT 4 SKIP 2" (3) "NEWS" (6)

(b) CONIT Query Examples

ACKNOWLDEMENT

I would like express my thanks to Miss Mieko Takao and Miss Miyuki Imai for typing the manuscript. This research is supported in part by the grant for Special Research Project on Formation Process of Information Systems and Organization of Scientific Information sponsored by the Ministry of Education, Science and Culture of the Government of Japan.

REFERENCES

[1] T. L. Kunii and H. S. Kunii, "Database Design", Proc. of Tenth Hawaii International Conf. on System Sciences, 200 (1977).

[2] D. T. Ross, "Structured Analysis (SA) : A Language for Communicating Ideas", IEEE Trans. Soft. Eng., 3, 16 (1977).

[3] M. W. Alford, "A Requirements Engineering Methodology for Real-Time Processing Requirements", IEEE Trans. Soft. Eng., 3, 60 (1977).

[4] D. L. Parnas, "On the Criteria To Be Used in Decomposing Systems into Modules", Comm. ACM, 15, 1053 (1972).

[5] D. D. Chamberlin and R. F. Boyce, "SEQUEL: A Structured English Query Language", Proc. ACM SIGFIDET Workshop, 249 (1974).

[6] T. L. Kunii and Y. Ishii, "The ADABAS Model, Part 1 Modelling of Logical ADABAS Architecture", (submitted for publication).

[7] D. Tsichritzis, "LSL: A Link and Selector Language", Proc. of International Conf. on Management of Data, 123 (1976).