1990

# Data-driven information management using computer aided systems engineering

Gautam Bose
*Lehigh University*

# Data-driven Information Management using Computer Aided Systems Engineering

by

Gautam Bose

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science

Lehigh University

1990

# Certificate of Approval

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science.

Date : May 11, 1980

Professor in Charge : Roger Nagel

Chairman of Department : Lawrence J. Vanneman

# Acknowledgements

I would like to thank Roger N. Nagel for his support and patience throughout my thesis work. His guidance and comments were very useful to provide continuity and direction to my discrete thoughts.

# Table of Contents

# List of figures

# Abstract

The objective of this thesis is to investigate the problem of information resource management in general and manufacturing information systems in particular. The thesis provides an approach to enterprise-wide, data-driven information systems. The issues which affect information management strategies are discussed, without getting into possibilities, which cannot be implemented without proven technology available in the market today.

The strategies at a conceptual level are first identified in the form of questions which discuss the need for that approach and also analyze the alternatives. The requirements of manufacturing information systems are then listed and implementation issues are discussed.

The use of Computer Aided Systems Engineering (CASE) tools to aid system development and maintenance is described. A survey of CASE tools available in the market today is included in the appendix section.

## 2. Introduction

There is a need for an enterprise-wide information management strategy. Information technology is no longer just a tool for efficiency. Consistent and accurate information must be available on-line throughout the organization, in order to use information as a competitive weapon.

Manufacturing operations are highly data intensive and most manufacturing companies have made strategic decisions to use information technology. However there is a lot of confusion in integrating the requirements of various manufacturing functions. A formal basis for developing information systems bottom-up is a pre-requisite for proper information sharing throughout the organization.

Data-driven systems are based on logical structure of data which is more stable than procedures which use the data. The five steps involved are (a) Specify business functions (b) Model business data (c) Show interaction between functions and data (d) Map data model to physical database tables (e) Write procedures to perform business functions using tables. Computer Aided Systems Engineering (CASE) encompasses the full systems life cycle. It provides formal techniques for engineering quality software systems.

## 3. Why Enterprise-wide Information Management ?

Few organizations today, can meet their business objectives without using some form of computerized information system. Successful organizations have recognized that accurate and timely operational, management and marketing information is key to success. An enterprise wide information management strategy sets direction for leveraging information technology based on business objectives, taking into account the present situation and future possibilities.

Increasing global competition has forced many corporations to become more competitive. Islands of information systems are implemented bottom up, without a common foundation or structure of the underlying data. This leads to incompatible systems as there is no consistency in the meaning of data throughout the organization.

Within an enterprise, each organization, and system has its own view of the data. Most businesses use inconsistent rules to control their data, which leads to information pollution. Not only are basic rules inconsistent, but they are also changing. There is a need for enterprise-wide, shared information systems which are fully compatible with one another. One way to make this happen is to use a single logical data dictionary which

acts as the information repository for all automated systems within the enterprise. Distributed database technology is available to make this possible. However a new approach to information systems is necessary.

Information management in many companies is similar to drinking out of a fire hydrant. The flow of information technologies is so strong that it is difficult to control consumption. Responding to these rapid changes in information technology requires a strong information management strategy. The migration from data processing to corporate information management creates a new role of information managers with responsibilities to manage corporate data models as an asset and using information resource as a competitive weapon.

The development of enterprise-wide systems are far from easy. There are many problems to overcome in terms of business, technologies, tools, techniques and the attitude of the users and existing data processing community. Business problems include the lack of a clear direction, no commitment from senior management, and multiple solutions to the same problem due to lack of control. The rapid evolution of technology has caused problems with the proliferation of micro-computers, short life span of computer hardware and a vast range of software.

## 4. Why Data-driven Information Systems ?

In any organization, the structure of data has the longest life followed by communication networks, operating system and database software, and computer hardware in that order. Logical representation of data is relatively stable, whereas procedures that use the data change frequently.

Therefore the logical data model, which reflects what the organization is, not how it currently operates, should be the base for system development. Information systems can be better integrated if data which are shared are controlled centrally by being part of the same logical data model.

It is no use trying to superimpose electronic information on to a sort of a tangled company procedures which has grown up around conventional paper based communications and islands of automation. Procedures in any organization is bound to change and information system development is still too labor intensive. Needless to say, most conventional data processing departments have huge application backlog, trying to keep up with changing procedures all the time.

The basic need is for an information-flow system which enables the information to be accessed in different forms to meet different requirements. At the heart of all

automated information systems is the struggle over "DATA". Information is "mined" from data. The real objective is to manage, store, give access to and provide the ability to manipulate and communicate the raw material of information and knowledge - "DATA".

Data driven systems have a formal basis and the program control structures are derived from the program data structures. This is key to flexible information systems. The relational data model provides ways to describe the data without any physical pointers to the actual storage of the data. Separating application programs from the physical organization of the data provides a high degree of data independence and operational flexibility.

Currently available computer hardware and software enables data to be processed quickly. Mainframe quality processing power of not to long ago are available on the desktop today at a fraction of the cost. This trend in hardware improvement is likely to continue with parallel processing machines in the future. The real problem is how to manage and coordinate the data that is being held and used in different ways. Automating adhoc procedures will not solve the problem in the long run. Information repositories must be developed to define the structure and constraints of the data once and use it consistently.

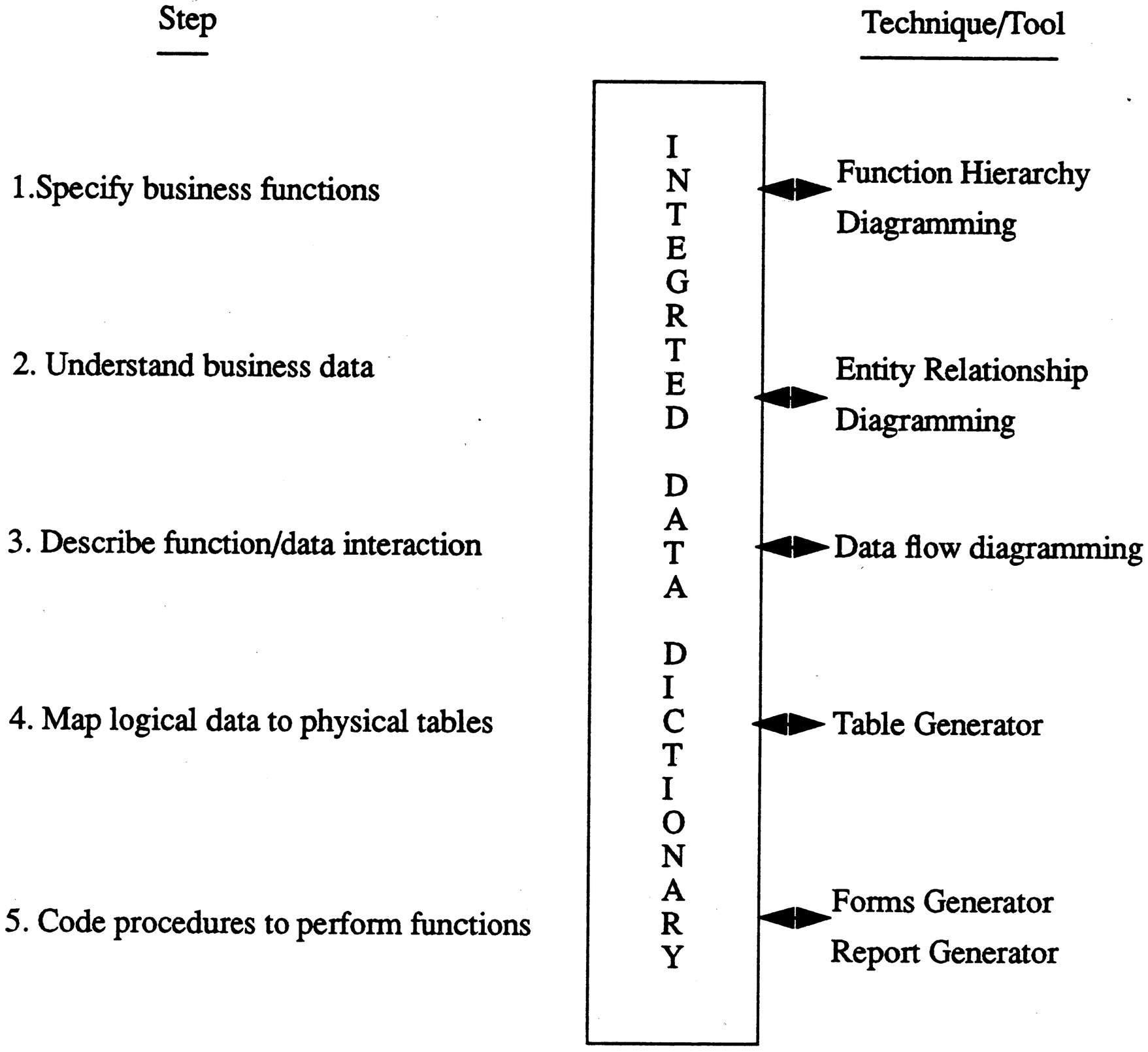


Fig.1 Data-driven Information System Development Methodology

## 5. Why Computer Aided Systems Engineering (CASE) ?

It is said that cobblers go around with holes in their shoes. The information processing profession is no better. Many computer information systems are built and maintained using crude tools and inadequate computer support. A large number of software project are never completed or arrive too late. Software systems need to be engineered and not crafted. There is a need to utilize active computer assistance to today's systems development efforts. Large information systems must not only be built but maintained and enhanced as requirements grow. Computer Aided Systems Engineering encompasses the full life cycle of systems development, and provides formal techniques similar to other engineering disciplines.

### Evolution of CASE

CASE is an evolving technology that promises automation of computerization. It provides useful tools to both end users and programmers for which there is a huge demand in the market. CASE technology hold promise for solving the worldwide software problem of too many applications and too few application developers.

The term CASE was first coined by Dr. John H. Manley in 1984 and originally stood for 'Computer Aided Software Engineering'. The first generation of CASE products

provided the systems analyst tools to draw, edit, update and store various diagrams used in the initial phases of the software development life cycle. These products came to be known as upper-CASE tools. Soon vendors of code generators got on to the CASE bandwagon. These tools automate the tasks in the later phases of the software life cycle and came to be known as lower-CASE tools. Lower-CASE tools support code generation, testing, debugging, compiling, prototyping and implementation. First generation CASE tools supported islands of automation in the systems development cycle but did not allow data and information to flow automatically from one phase to another.

The second generation of CASE tools began appearing in 1986. These products allowed information from the requirements phase to develop specifications. This information in turn, is used automatically in the design and build phases. Second generation CASE products provided tool integration by means of a central storage facility or information repository. Commercially available Relational Database Management Systems (RDBMS) were used for CASE repositories by many CASE vendors. Real-time programming techniques and Artificial Intelligence (AI) technology are being added to current CASE products. With AI, diagrams could be replaced by logic models from which runtime systems could be generated automatically.

Tools that deal with up-front strategic and management issues of software development are categorized as "pre-CASE" tools. Management functions supported by pre-CASE tools include project planning, scheduling, and estimating. With database technology established as part of the CASE body of knowledge, developers need tools to transfer existing code into the CASE environment. This is the "post-CASE" process of converting sphagetti code and storing it in CASE repository. Integration of various tools has given a new meaning to CASE. CASE helps produce better systems and not just code. The term CASE is thus more commonly known as "Computer Aided Systems Engineering".

Applying formal techniques to the complex, time-sensitive requirements of information systems is not practical without automated tools. Automated tools impose formality, increase the speed at which systems can be built and modified, and most importantly, co-ordinate the vast amount of knowledge that must be collected and updated.

CASE technology promises to be the delivery vehicle of enterprise-wide information repository. Though there is no standard for corporate repository as yet, there is a trend to consolidate corporate data centers into so call super-data centers. This will ease the task of creating single data dictionaries using distributed databases.

4.Pre-CASE 1.Upper-CASE 3.Lower-CASE 5.Post-CASE

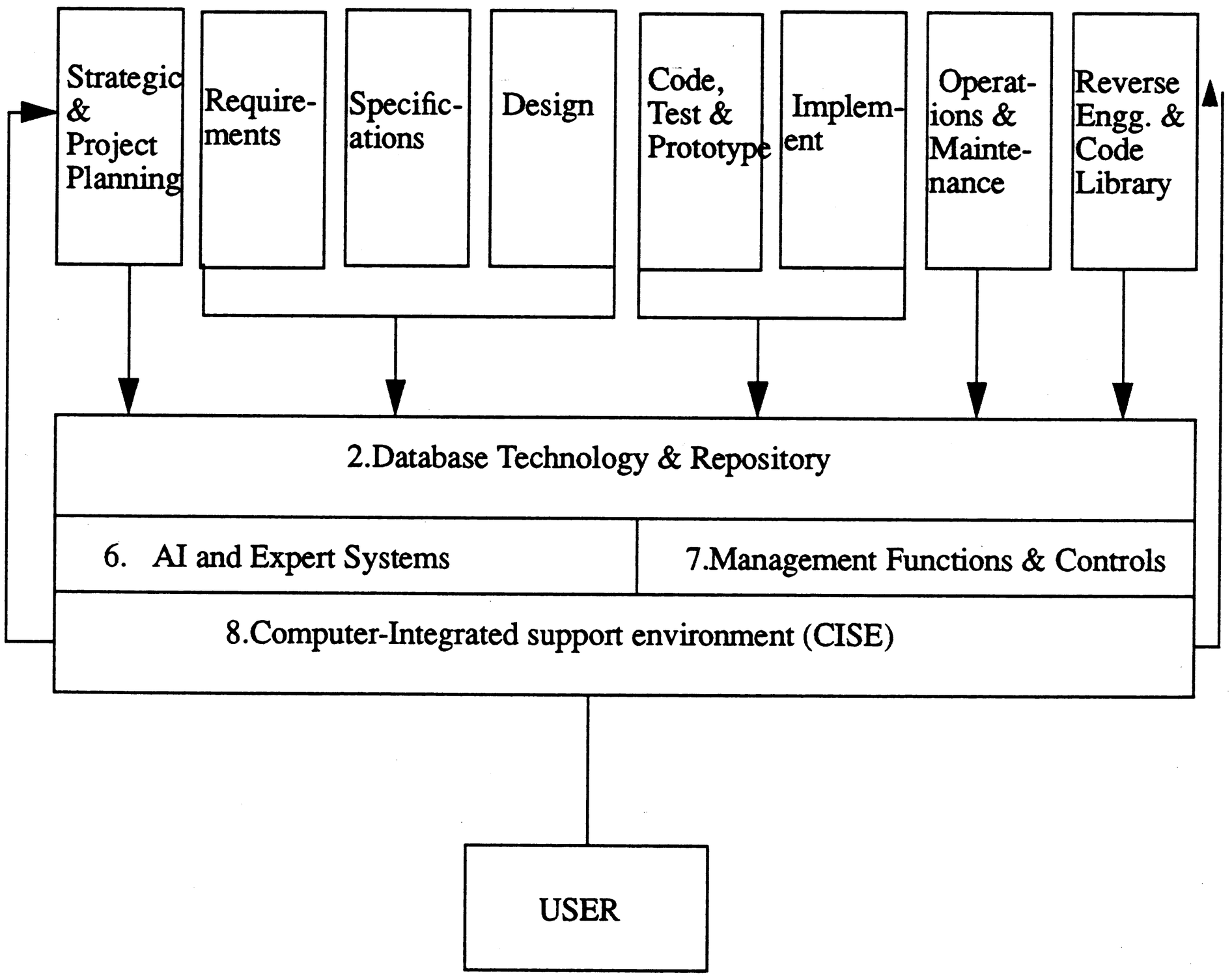


Fig. 2 Evolution of CASE

The most important CASE tool is an active, on-line, multi-user data dictionary. The dictionary should hold and manage all the information collected and derived during the development and production running of a business system. The dictionary provides the vital integration between the different tool sets used in the various phases of the system development life-cycle. The content of the design dictionary reflects all the complex interdependencies and relationships of a business problem in a form which can be clearly understood. When the business changes from the specification in the design dictionary, it may be updated to reflect the new situation and then used to indicate how the production should be amended to incorporate the current business functions. When the system is being built, the dictionary enables the implementations to be recorded and linked to the design and business models. A by-product should be complete, up-to-date and fully cross-referenced system documentation. Good documentation is one of the keys to any successful system.

A data dictionary is used for storing the meaning of all data within the enterprise, across all application systems. In other words it stores data about data or meta data. Within an enterprise, each organization, system, person, and function has its own meaning of the data and business rules. In order to consistently define, co-

ordinate and manage the meaning of the business data, it is required to store them in a central information repository. A data dictionary is used to drive a data driven information system and hence is a very key tool.

The data dictionary is more than a catalog containing a simple list of data elements. It must store the description of all business entities. It must list their attributes, data types, default values and other constraints. Relationships between the different entities must be stored in the context of each application. The dictionary must also store referential integrity information and master detail relationships. For example, each purchase order may be composed of one or more purchase order items. In case the purchase order is cancelled, all the detail order items must be automatically deleted.

<u>CASE systems must provide the following features</u> :

1. Tools to assist every phase of the system life cycle.
2. Integrated, multi-user, shared information repository for storing all system information.
3. Graphics interface for drawing structured diagrams.
4. Prototyping and database sizing tools.
5. Automatic code generation for screens and reports.
6. Design checker and analyzer.
7. Application maintenance and version control facilities.

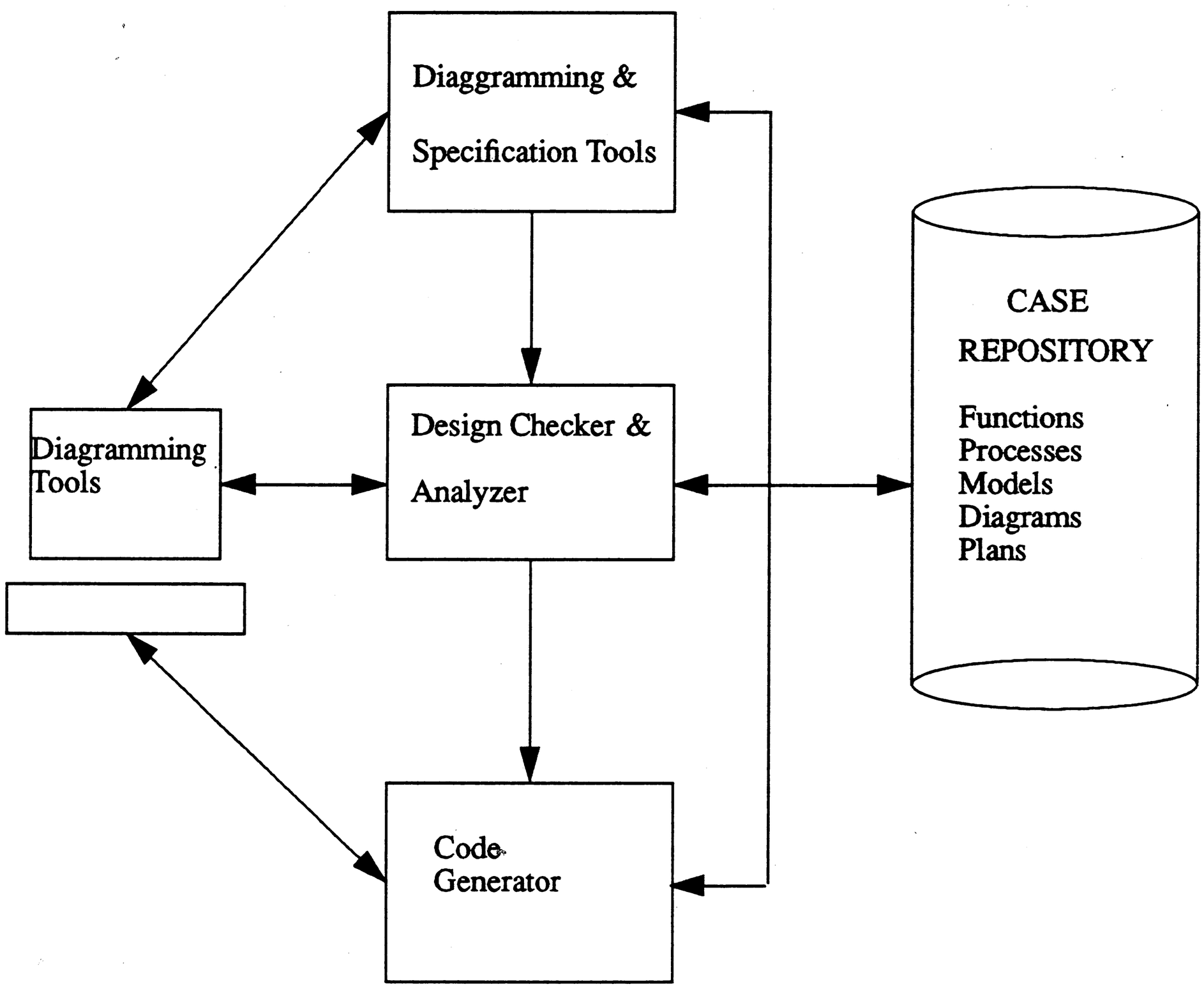


Fig. 3. Integrated CASE Environment

## 6. Why Logical Data Modelling ?

Data modelling is one of the important activities of data driven system design. A logical data model is basically the structure of information used in the business. It is the whole infrastructure of rules that comprise the concept of the business. Logical Data models provide a formal basis for tools and techniques used in developing and using data driven information systems.

With respect to database design, logical data models provide a representation of the application that captures the static and dynamic properties needed to support the desired transactions and queries. In order to build a system that satisfies some functional need, it is required to explicitly and unambiguously state the set of business rules which define the function. In data driven systems, logical design must be separate and precede physical design.

Choosing an appropriate information modelling tool for defining the conceptual schema is very important. The tool must have a semantic form and a graphic form for visualization. Several methodologies exist for constructing the business data model. Any such methodology should satisfy the following basic criteria :

1. All the rules are consistent.
2. It is extendible, ensuring that new rules do not invalidate or conflict with each other or with those already in the conceptual schema.
3. It is accessible, in that all of its rules are easily understandable to all who use it.
4. It is transformable, in that its internal rules can be algorithmically transformed into software.

## 7. Why Entity Relationship Diagramming ?

Entity relationship diagramming is a technique to represent the logical relationships that may exist between various entities in a business environment. Originally invented by Peter Chen, this methodology has been extensively enhanced to provide more functionality. In 'Rule Based Entity Modelling' simple English language statements are used to capture the business rules which constrain the interaction between various entities in the system.

Entity Relationship Diagramming provides a way of defining the things of significance or entities which concern the enterprise and the relationships among them. It also provides an accurate model to describe the structure of the information used by the enterprise.

Some definitions and conventions:

An 'Entity' is anything of significance about which information is to be known or held. It is any named thing which has a seperate and distinct existence, either real or imagined. Examples of entities are part, `department, employee, person, building, purchase order, account etc.

An 'Attribute' is something to be recorded about an entity. It is any detail which serves to qualify, identify, classify, quantify or express the state of occurrance of an entity. Each entity can have one and only one occurrance of each attribute. Examples of attributes of 'part' entity are part number, description, unit of measure, price etc.

A 'Relationship' is any significant way in which two things are associated. In other words it specifies what one entity has to do with another.

Entities are shown in soft boxes (with rounded corners), distinguished from logical tables shown in hard box. Entities are mutually exclusive and always singular (Eg. 'part' is an entity but not 'parts'). A sub-entity is an entity whose occurrances are also examples of another entity. A sub-entity concept is different from entity type. Descriptive attributes for entities are sometimes shown as unique(#) identifiers, mandatory(*) or optional(o).

Relationships that add uniqueness to an entity are marked with a crossbar. Dashed lines indicate optional relationships; undashed, mandatory relationships. These modelling techniques drive primary and foreign key implementation and relational database column/record field design. Relationships are labelled at both ends to show unambiguous subject/object orientation. Relationships are shown with the trident or crow's foot symbol to indicate multiple relationship degree. Multiple relationships can occur between entities, and relationships can occur between an entity and itself.

Arcs are used to show exclusive relationships. Either relationship may be true for an occurance of an entity, but not both. These techniques ensure that effective implementation options can be weighed during data design. Cross-checking techniques ensure the model's accuracy and completeness.

An Entity Relationship Diagram is basically a structural representation of the business rules. It is simple yet powerful enough. A solid line represents a mandatory or MUST BE relationship. A dashed line represents an optional or MAY BE relationship. The following is the reading rule for the sample ER diagrams shown below :

EACH

<entity name 1>

MUST BE

or

MAY BE

<relationship name>

ONE AND ONLY ONE

or

ONE OR MORE

<entity name 2>

Consider the data model shown below in figure #, which captures the following business rules:

1. Each part category MAY BE the classification for ONE OR MORE part.
2. Each part MUST BE of ONE AND ONLY ONE part category.
3. Each part MAY BE made via ONE OR MORE work order.
4. Each work order MUST BE for production of ONE AND ONLY ONE part.
5. Each work order MUST BE carried out in ONE OR MORE work center.
6. Each work center MAY BE used for ONE OR MORE work order.

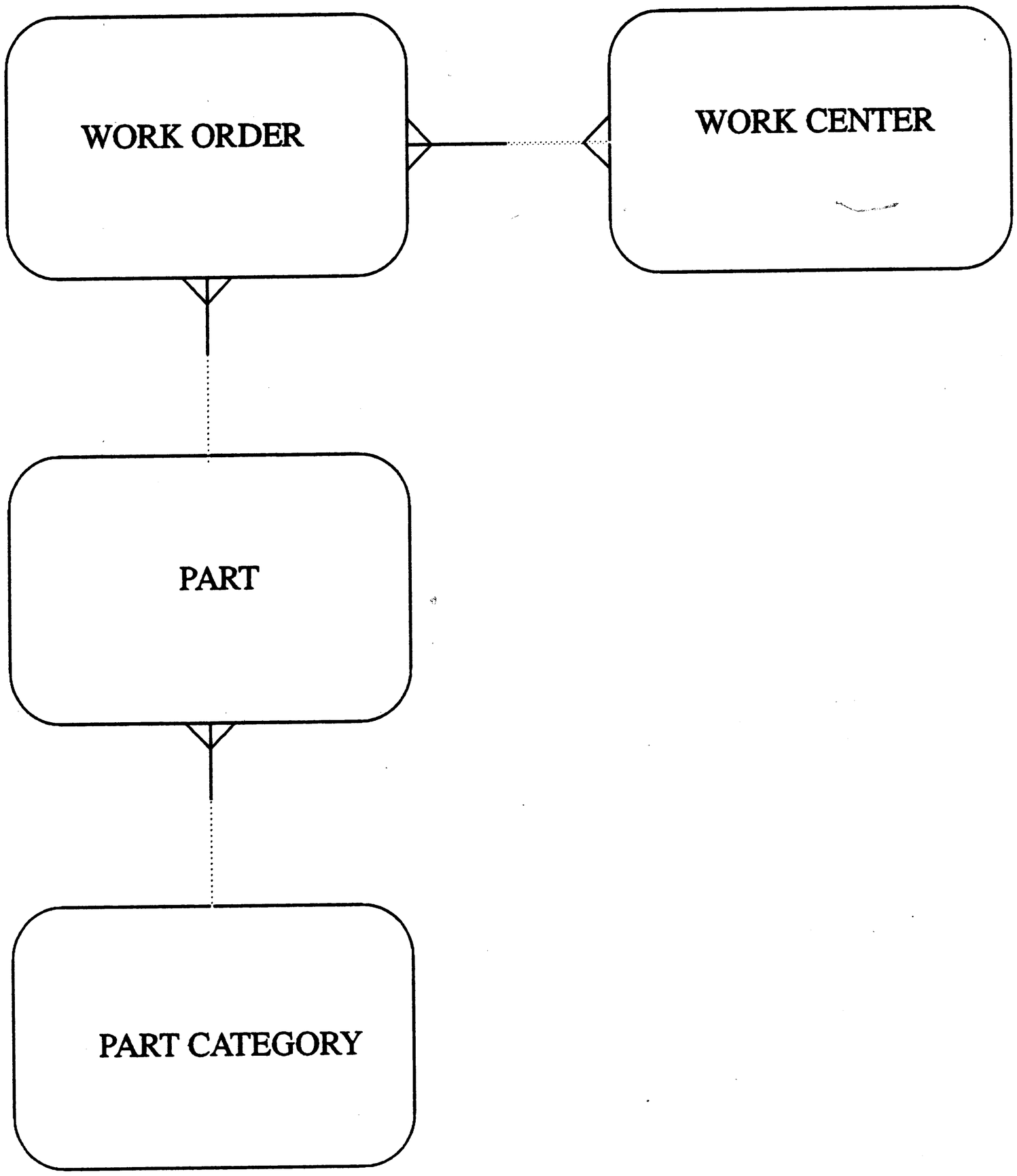


Fig. 5 Entity Relationship Diagram

## 8. Why Relational Data model ?

The most important reason for choosing the relational data model is data independence, which is defined as the immunity of applications to changes in the physical and logical environments. The relational data model also grew out of a need for representing data in a well defined, rigorous manner; to improve management of large volumes of data; ensure proper data sharing; enforce standards; and improve integrity and security.

One of the main attractions of the relational model is its mathematical clarity, which facilitates the formulation of a precise data manipulation language. The common data models at that time were hierarchical, inverted list, and network. Applications developed using these models required knowledge of the data organization. Furthermore access paths were built into the logic and code. Programs written in COBOL, FORTRAN or PL/1 required operating system storage structures. Any change to the way the data was physically stored (e.g block sizes or field layouts) required extensive software maintenance and program modifications.

Data independence means freedom of applications from data access strategies such as groupings or orderings. In the relational model based on set theory, there is no

concept of ordering of elements within a set. Because the order of attributes in a relation is not significant, new attributes can be easily added without affecting existing applications. Non-relational systems often require application developers to have prior knowledge of physical ordering of records.

Non-relational systems represent relationships between groupings of records with logical pointer chains. Pointer chains must be built, followed and understood by the application developers. Any changes in the relationships of the groupings would affect the pointer chains and therefore all application programs following those chains. The relational model does not rely on any chaining but instead bonds the tables by the values of the foreign keys. Relational systems lead to significantly reduced costs for developing new applications when compared to network or hierarchic approaches.

With the simplicity of the relational model, only set operators need to be used to report or manipulate data. Because there are no data access path restrictions, data bases can be designed based on the semantics of the data instead of by access requirements as is done in procedural systems. Because the structure of a relation allows tuples to be in any order, access to data can be non-procedural.

Non-procedural access is requesting what data is required and not how it is obtained. This makes access to data much easier and more flexible.

Another important reason for the relational data model is ease of data distribution. Due to the simplistic structure of relational data model, data can be physically distributed on multiple computers without affecting application programs. This is possible because there is no navigational knowledge embedded within the application programs.

## Structure of the Relational Model

The relational model was proposed by E.F Codd in 1970. He outlined the relational nomenclature, definitions and theory. Data in the relational model is developed around two-dimensional tables called relations. Relations correspond to entities in the real world. Entities are things of significance which can be distinguishable and uniquely identifiable. Each row of the tuple is called a tuple, and each column of the table is called an attribute. Values of attributes are chosen from domains or ranges of acceptable values. Consider the relations shown in tables 1 & 2 . The relation or table name SUPPLIER contains six rows or tuples. Each tuple consists of four attributes or columns : SNUM, SNAME, SCITY, and SREGION. The relation

REGION consists of three attributes : SREGION, DESC and QUOTA.

| SNUM | SNAME | SCITY | SREGION |
|---|---|---|---|
| S1 | AAA | Long Beach | SW |
| S4 | ZZZ | Anaheim | SW |
| S2 | DDD | New York | NE |
| S12 | DEN | Denver | CN |
| S34 | ACME | Long Beach | SW |

Table 1 : Relation SUPPLIERS

| SREGION | DESC | QUOTA |
|---|---|---|
| SW | Southwest US | 1,000K |
| NW | Northwest US | 2,300K |
| NE | Northeast US | 4,400K |
| CN | Central US | 3,400K |

Table 2 : Relation REGIONS

Data Manipulation Languages (DML) support processing of the data structure. The relational model uses PROJECTION and JOIN to report or extract data from relations. PROJECTIONs remove attributes that are not required, yielding a new relation with the remaining columns. JOIN allows two relations to be joined over a common attribute, thus yielding a new relation. There are a few other relational set operators such as SELECT, UNION, INTERSECT, PRODUCT, DIFFERENCE and DIVIDE.

The relational model requires two data integrity rules: entity integrity and referential integrity. Integrity rules are tightly coupled with the concept of primary and foreign keys. Primary keys are unique tuple identifiers, and are made up of the minimum number of attributes such that no two rows of the table contain the same values for that primary key. Primary keys serve as record identifiers or unique identification function. For example in the SUPPLIER relation in table 1, if SNUMs are guaranteed to be unique and are never null, then SNUM can be a primary key. A foreign key is an attribute or combination of attributes in one relation, whose values are used to match those of the primary key of another relation. Foreign and primary keys represent references or logical relationships that hold the relations together.

Entity integrity states that no attribute participating in the primary key of a relation is allowed to accept null values. In addition, none of the attributes that make up the primary key can be discarded without destroying the uniqueness property. By including semantic aspects through the use of keys, the relational model provides more than just syntactic construct.

Referential integrity rule states that every value of a foreign key in any relation must exist as a value of a

primary key of a another relation or be null. For example in the SUPPLIER relation, SREGION is a foreign key and all non null values of SREGION in the SUPPLIER relation must exist in the set of values of SREGION in the REGION relation shown in table 2.

## 9. Why Three Schema Data Management Architecture ?

Data structures employed in an integrated data base management system must address three goals : enterprise support, user support, and machine access for retrieval and storage.

Enterprise support requires logical completeness. If data have been gathered and maintained at considerable cost, then it is essential that it be possible to use this data to respond to any logically meaningful query.

User support requires logical simplicity. Regardless of the complexity of the structures needed to support the enterprise's data processing, it is essential that the structures with which an individual user must interact be both simple and well suited to his/her needs.

And, for machine access to the stored data, data description must be provided at a level low enough to permit efficient operation by the physical devices.

Unfortunately, these requirements are somewhat incompatible. A structure that is logically complete enough for the enterprise is not convenient for end users or programmers. Likewise, a structure that is well designed for one application may not be suitable for another.

A promising mechanism for resolving these difficulties is the three schema model proposed by ANSI/SPARC as shown in figure 5. Rather than attempt to define a single class of data structures of universal applicability, ANSI/SPARC proposes three levels of structures, one each for the enterprise, the users and the machine itself. Such a model requires not only the ability to declare data structures of different classes, but to define maps between these structures.

ANSI/X3/SPARC three-schema architecture as the data management strategy has clear long term advantages. It defines three seperate but related levels of data base schema : external schema, conceptual schema and internal schema.

The conceptual schema must be complete. It supports the enterprise and its view of the data required for its operations. The conceptual schema may be mapped to one or more external schemata.

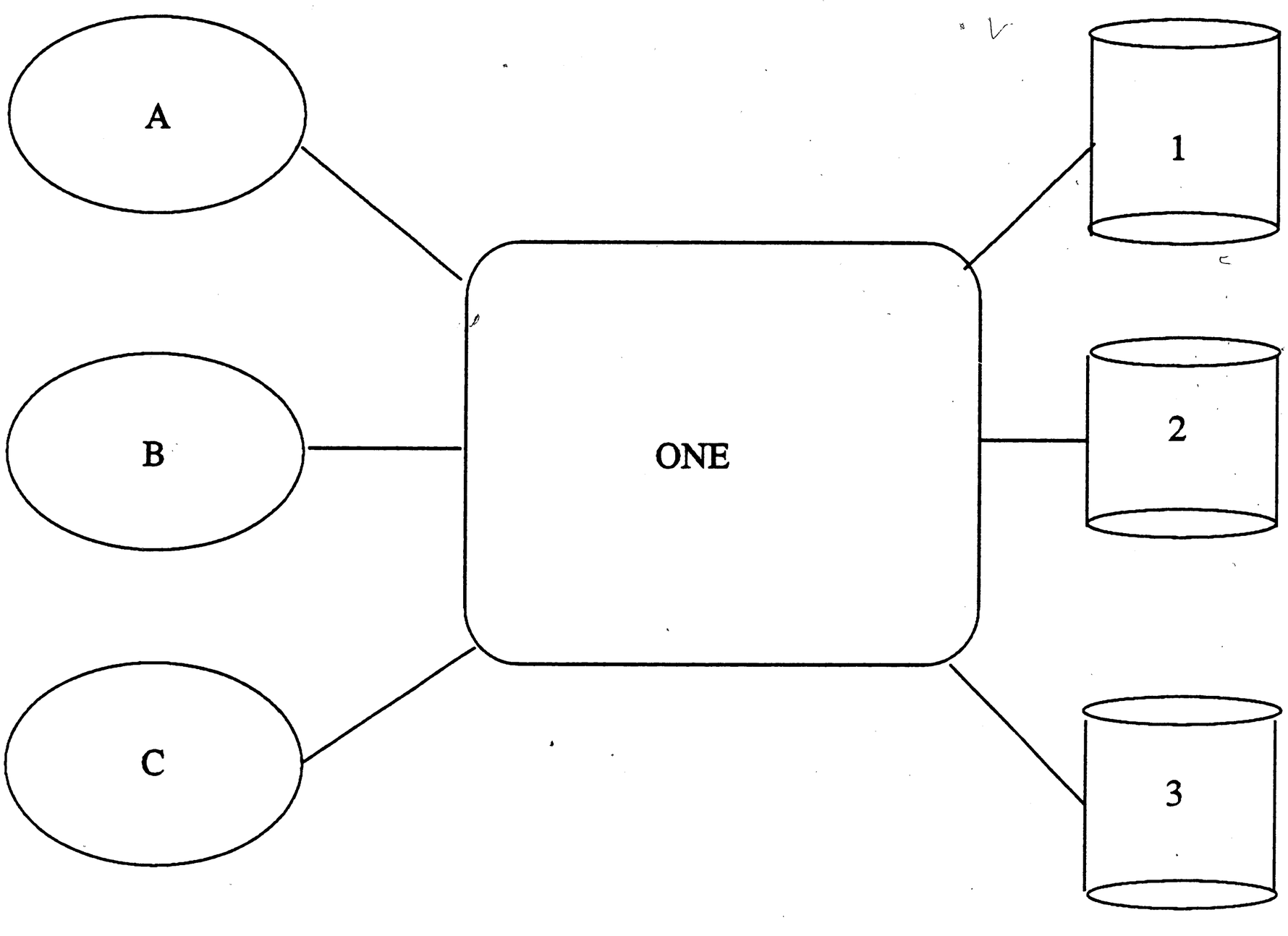


Fig. 4 Three Schema Architecture

The internal schema is needed for access and storage of data on physical devices. The details of the internal schema is best left to the database management system software as issues like optimization, security and rollback are too complicated for the average programmer. There is no point in re-inventing the wheel as there are many database management systems which does the job sufficiently well for most applications.

Unlike the traditional two-schema approach, which maps external schema to database files, the ANSI concept, through the introduction of conceptual schema, makes external and internal schemata independent of one another. The conceptual schema drives the Information Resource Management planning and the implementation from the data rather than the application or process perspective. Thus these methodologies are called data driven.

The planning procedure builds on the conceptual schema - called the data architecture - from the top down. The implementation procedures build on the conceptual schema - called the normalized data model - from the bottom up. They both employ the same information modelling tool for describing the conceptual schema.

Entity Relationship Model is well suited for representing the conceptual schema in the three schema architecture. The Relational model is powerful, yet simple for use as external schema. It provides a fourth generation query language - SQL. The internal schema depends upon the implementation of the specific Data Base Management Systems available in the market place today and can be balanced B-TREE, network CODASYL etc.

## 10. Why Relational Database Management Systems (DBMS) ?

A database management system is a set of programs that insulate the application programs from the physical disk files.

However a DBMS was not invented to replace the disk management software of the computer's operating system, most of which feature at least two file access modes - sequential and direct. Most operating systems ignore other types of access and ignore such services as data dictionary management, use of query languages, backup/recovery etc. The DBMS is intended to solve just those problems and is a complement to the file management features of the operating system. But many users use a DBMS to manage their disk files and call the collection of files 'database'.

Unfortunately a database is not the consequence of the use of a DBMS. A database provides a means of answering all expected or unexpected questions of decision-makers regarding their data. A database is an information and communication tool. The value of decisions that rely on facts from a database is often strategic. In order to do this, the database itself needs to do the following :

1. It must be exhaustive in order to answer all potential questions.
2. It must be consistent in order to contain accurate information.
3. It must be structured in order to allow for the retrieval of vast amounts of data.

An on-line shared database provides horizontal integration. The central database is common to all departments and updated constantly. People can communicate better by feeling the progress of the work of other users and hence can make better decisions.

A database provides a means for vertical integration as managers can directly access the database directly and verify small facts. In addition, everybody shares the same database and has the same raw material to work with. An on-line database also eliminates costly paper procedures and delays.

## 11. Manufacturing Functions and their data requirements

Most manufacturing functions are very data intensive. Each stage of material transformation, requires and also generates a lot of data.

Like any other data-driven system it is first necessary to understand the business functions and the structure of the data used by those functions. The data structure is then logically represented using techniques like entity-relationship diagrams. The logical data model is then mapped to physical database tables with the aid of CASE tools. Procedural programs are then implemented to carry out the business functions.

Generically speaking, the major manufacturing functions are [2] (a) Marketing and sales (b) Manufacturing data preparation (c) Production planning and inventory control (d) Production scheduling (e) Process supervision and (f) Quality Assurance

### a. Marketing and Sales

Marketing and sales provide the primary interfaces between manufacturing facility and its customers. They inform customers of available products, generates orders for selected products, price those products, negotiate delivery schedules, track shop floor performance in meeting

those schedules and ensure customer satisfaction after delivery. They also conduct needs analysis to determine potentially profitable new products.

To perform the above functions, marketing needs to access multiple databases. These include product catalogs, customer orders, both current and projected manufacturing capacities, finished products inventory, schedules and anticipated completion and delivery times.

b. Manufacturing Data preparation

Manufacturing data preparation encompasses all the functions required to generate the data needed to manufacture a product capable of meeting customer requirements. These functions include engineering and process planning.

Engineering translates a set of customer requirements into product designs which include detailed 3-D drawings, geometry data, tolerances and other required manufacturing specifications. All these design data is then used by process planning to generate a complete bill or materials, tools, machines, fixtures and precise machining instructions used for the entire fabrication process.

The data structures are most complex in this domain. First, specifications are used to construct computer aided (CAD) design drawings. Depending upon the sophistication of the CAD tools used, additional data may be required to for generating process plans from the design data.

Process planning basically develops a sequence of operations to meet the design requirements and describe the processes that are needed to perform each of these operations. This itself is a complex task and requires a lot of expertise. First, the product is given a group technology classification code. This code is then used to retrieve existing plans for products with similar processing requirements. Sometimes process plans are generated using some algorithms.

c. Production Planning and Inventory Control

Production planning is responsible for developing a list of jobs to be executed on the shop floor during the next planning horizon. In addition it determines the machine capacities and raw materials necessary to do these jobs.

Aggregate production planning (APP) uses the current and projected demands established by marketing to set production quotas and inventory requirements for each

product type during the chosen planning horizon using small time buckets (usually one week). These inventory requirements include all raw materials, tools, fixtures, castings, forgings etc. Aggregate production planning also continually monitors and updates production quotas and inventory policies based on marketing or other feedback sources.

Aggregate production planning must access the marketing data to know the actual and forecast demands. The process planning data is required to know the processing requirements for each of the products that make up the demand. It is also required to know (a) current inventory status on tools, finished goods, work-in-process, raw materials and available shop floor capacities. Data about cancelled or changed orders is also accessed.

Detailed production planning uses the above assigned quotas to generate production jobs for each time period depending upon available resources. They explode the Bill of Materials (BOM) for each product and check the inventory status of each raw material. Detailed production planning provides shop floor release dates and availability of orders. Sometimes jobs may be sub-contracted to external vendors to meet customer deadlines.

Detailed production planning must access the databases updated by aggregate production planning. It is also required to access the process planning databases containing the process durations and precedence relations for each product to be produced. Detailed process information like machine uptimes, preventive maintenance etc. are also required.

d. Production Scheduling

Production scheduling develops detailed schedules of the operations required to complete the jobs issued by detailed production planning. These operations are then assigned to the various processes together with their anticipated start and finish times.

Once a production schedule has been generated, it is necessary to co-ordinate activities at each process to ensure that the schedule is met. This inter-process coordination requires monitoring of the feedback from process supervisors. This feedback ensures that all required materials have arrived at the process before the stated start times and to determine if the process will finish its assigned tasks at the anticipated finish times.

The production scheduler is responsible for maintaining an accurate schedule of activities at all

processes on the shop floor. That schedule must be updated whenever (a) a new job list is received (b) an existing job is cancelled, finished or given a priority update, and (c) a process experiences an unexpected delay. When a new job is assigned, the process scheduler must first determine all of the required operations and the processes.

The process scheduler needs to retrieve both the job list created by the detailed production plan and the process plan for each entry on that list. The current schedule is then updated by inserting the start and finish times for these new operations. This requires accessing the current schedule and process utilization databases and executes some scheduling algorithm. Similar tasks are required for priority updates and unexpected delays.

e. Process Supervision

The process supervisor has to implement the precise instructions from the process plan for every assigned operation and also monitor the process during its execution, to verify conformance to those instructions. To do this several databases needs to be accessed : List of assigned jobs and scheduled start and finish times and their associated process plans, Numerical control code libraries, part description data, tool and fixture data. After each job has been completed, the supervisor must

update the job database indicating the exact operations performed, the total processing time in the scheduling database, and the equipment and tool usage in their respective databases.

f. Quality Assurance

Quality assurance has two major functions. First, it verifies that the output from each process meets the specifications prepared by engineering. This is done by both in-line and off-line inspections. Second, it keeps historical records which can be used to improve the quality of all phases of the manufacturing system. This data is also used for machine maintenance and tool replacement.

Quality assurance requires access to inspection plans, equipment usage charts, machine maintenance schedules. Information is archived regarding each product. This includes CAD designs, process plans, inspection and machining procedures, and material used.

## 12. Implementing Data Driven Systems

Implementing data driven systems requires going through the full systems life cycle methodology. The system is driven by the entity relationship data model, which represents the corporate data structure. The business functions are represented by the function hierarchy diagram. A business function is essentially a process of transforming the data and this interaction between the data and functions is represented by the data flows. Unlike procedure oriented systems however, the data flows do not drive the system. A generic essential data flow diagram is shown in figure 6.

An system's life cycle is defined as the interval of time between its conception and its replacement. It encompasses the following stages - Strategy, Analysis, Design, Build and User documentation, Transition and Production.

The Strategy phase determines what needs to be done and how much it should cost. It results in a statement of business needs, objectives, priorities, and constraints along with a definition or scope for the system development project.

Structured interviewing and data modelling techniques are used and presented to the user using visual techniques. One modelling technique - the function hierarchy diagram is used to identify the structure of the functions of the organization, while a second called the entity relationship diagramming is used to identify the structure of its information.

The analysis phase specifies the exact user requirements. It results in detailed statements of user specifications, including definition of business functions involved. These are derived from additional interviews and detailed documents.

The design phase decides what the systems should look like and what tools and components should be used. It results in a detailed hardware, software and physical file structures used to fulfill the above requirements.

The build phase results in a tested system complete with defined procedures. The user documentation is developed simultaneously. The transition phase moves the organization from its present system to a new one. The production phase take the system live and is marked by the operation of a smoothly running system, with planned, non-disruptive maintenance and future enhancements.

## 14. Conclusion

There is an immediate need for information systems management in most corporations. Growing applications backlog, slow labor intensive and unpredictable systems development, error ridden software, inflexible procedure oriented systems and large software maintenance effort are some of the common problems.

Large software systems cannot by crafted. They must be engineered with a formal basis. Data-driven system development approach supported by a good Computer Aided Systems Engineering (CASE) methodology is a promising direction for enterprise-wide information systems.

Manufacturing information systems are highly data dependent. Procedures and strategies constantly change to keep up with the market conditions. Flexibility in physical factories alone will not be sufficient. Information systems which support the manufacturing operations also have to be equally flexible. Non-procedural, relational systems can support the adhoc queries and reports required to make good decisions.

Distributed database management systems technology is available today. It is possible to define and store a single copy of the data and share it throughout the

organization. This will ensure consistency of data and the most current information to be available to everyone. Paperless procedures and electronic data interchange is possible only when there is data consistency and integrity throughout the corporation and its divisions.

Information Technology is finally getting more importance in many corporations. The position of Chief Information Officer reporting directly to the President of the company is becoming a reality. The migration from data processing to corporate information management has created a new role of information managers with responsibilities to manage corporate data models as an asset and using information resource as a competitive weapon.

## 15. Bibliography


1. Appleton Daniel, S. "The modern data dictionary" Datamation, March 1, 1987

2. Codd, E.F. "Relational Data Base: A Practical Foundation for productivity." - Communications of the ACM, Feb.1982

3. Date, C. J. "An introduction to Database Systems" Addison-Wesley, 1984

4. Teorey, Toby J., Yang, et. al. "A logical design methodology for Relational Databases using the Extended Entity Relationship Model" - ACM Computing Surveys, June 1986

5. Barker, Richard "Entity Relationship Modelling" Addison-Wesley Publishing Company, 1989

6. Computer Aided Software Engineering Symposium, Fall 1987

7. Puttre, Micheal et. al "Repositories get down to business" - Information Week, May 29, 1989

8. Davis, Wayne et. al "Data Management Strategies For Computer Integrated Manufacturing Systems"

9. Codd, E. F "Is Your Relational Database Management System Really Relational ? "

10. Davis, Jack M. "The Evolution of CASE - Automation of Computerization " Oracle Magazine, Fall 1989

11. Meister, A. E. "Database considerations for Computer Integrated Manufacturing" National Computer Graphics Association 1987.

12. Daniel S. Appleton "Information Resource Management and The Factory of the Future" 1984

13. Daniel S. Appleton "Data-driven Prototyping"

# APPENDIX A

## Survey of CASE tools available today

CASE tools available today can be categorized into three types. They are :

1. TOOLKIT

Sets of integrated tools that support one type of software development function or job class.

2. WORKBENCH

General purpose software development environment supporting the full range of software job.

3. METHODOLOGY COMPANION

Computerized assistance for a particular software development methodology.

Product Name : ADPAC CASE TOOLS

Vendor Name : Adpac Corporation

Category : Toolkit

Primary Use : Systems wide data analysis, reporting, diagramming and normalization.

Primary User : Programmer analyst, data administrator, systems development manager

Toolkits : Graphics, design, expert system

Methodologies : All Generic methodologies

Repository ? :

Price : $42,000 (mainframe)

Product Name : ANATOOL

Vendor Name : Advanced Logical Software

Category : Toolkit, Methodology Companion

Primary Use : Analysis stage

Primary User : Analysts, system engineers

Toolkits : Analyst

Methodologies : Yourdon

Systems : Information

Price : $925

Product Name : MULTI/CAM

Vendor Name : AGS Management Systems Inc.

Category : Toolkit

Primary Use : Integrates CASE methodologies into micro/mainframe environment

Primary User : Project managers, Programmer/Analyst

Toolkits : Entire CASE Library

Methodologies : SDM/Structured SDM/Standard

Repository ? : No

Price : $97,000

Product Name : AION DEVELOPMENT SYSTEM

Vendor Name : Aion Corporation

Category : Toolkit

Primary Use : Expert System Application Development

Primary User : Analysts and Programmers

Toolkits : Development Environment

Methodologies :

Repository ? : Yes

Price : $95,000 (mainframe)

Product Name : Life-Cycle Productivity System

Vendor Name : American Management Systems

Category : Workbench

Primary Use : Integrated set of productivity tools for all phases of development & maintenance

Primary User : Analysts, designers, programmers

Toolkits : For each phase of system development

Methodologies : SDM, Spectrum, Method 1

Repository ? : Yes

Price : $16,000 (PC)

Product Name : CORVET

Vendor Name : Analysts International Corp.

Category : Workbench

Primary Use : Support of design through Implementation and maintenance

Primary User : Analyst, programmer

Toolkits : Design, screen, report and code generator

Methodologies :

Repository ? : Yes

Price : $75,000

Product Name : PROJECT WORKBENCH

Vendor Name : Applied Business Technology

Category : Toolkit

Primary Use : Plan and control projects

Primary User : Group managers, project managers

Toolkits : Project Scheduler, Project Tracker

Methodologies :

Repository ? : NO

Price : $1150

Product Name : FOUNDATION

Vendor Name : Arthur Andersen

Category : Workbench

Primary Use : Plan, design, implement and support transaction processing systems

Primary User : Designer, Implementor, Maintainer

Toolkits : Design, Implement, Test and Config Mgt.

Methodologies :

Repository ? : YES

Price : $200,000

| | |
|---|---|
| Product Name | : INSTALL/1 |
| Vendor Name | : Arthur Andersen |
| Category | : Workbench, Methodology Companion |
| Primary Use | : Support for detailed design, coding & testing |
| Primary User | : Implementor, support specialist |
| Toolkits | : Code generator, test |
| Methodologies | : METHOD/1 |
| Repository ? | : YES |
| Price | : $200,000 |

| | |
|---|---|
| Product Name | : DESIGN/1 |
| Vendor Name | : Arthur Andersen |
| Category | : Toolkit, Methodology Companion |
| Primary Use | : Support integrated systems design, screen prototyping |
| Primary User | : Systems Analysts, Project Managers |
| Toolkits | : |
| Methodologies | : METHOD/1 |
| Repository ? | : NO |
| Price | : $7000 (PC) |

Product Name : STRIAD

Vendor Name : Arthur Young

Category : Workbench

Primary Use : Consulting support

Primary User : Systems Engineers

Toolkits :

Methodologies : AY/CSM, AY/IEM

Repository ? :

Price :

Product Name : THE DEVELOPER

Vendor Name : Asyst Technologies, Inc.

Category : Toolkit

Primary Use : Support DP professionals

Primary User : System Analysts

Toolkits : Analyst, Designer

Methodologies :

Repository ? : NO

Price : $4900

Product Name : SOFTWARE BACKPLANE

Vendor Name : Atherton Technology

Category : Toolkit, Workbench

Primary Use : Integration of applications

Primary User : Managers, programmers

Toolkits :

Methodologies :

Repository ? : YES

Price : $75,000

Product Name : DESIGN GRAPHICS SYSTEM

Vendor Name : The CADWARE Group

Category : Toolkit

Primary Use : Support system analysts

Primary User : System/Business Analyst

Toolkits :

Methodologies : IDEF

Repository ? : NO

Price : $1995 (PC)

Product Name : PACBASE

Vendor Name : CGI Systems, Inc.

Category : Code Generator Workbench

Primary Use : Automate Full Life Cycle CASE system

Primary User : Development Center Staff

Toolkits : PC Analyst/Programmer and End-User query

Methodologies : Merise, Yourdon and others

Repository ? : Yes

Price : 140,000

Product Name : SchemaGen, ER-Designer

Vendor Name : Chen & Associates, Inc.

Category : Data design toolkit

Primary Use : Generating data schemas for a given DBMS

Primary User : Database Designer

Toolkits : E-R Modeller package

Methodologies : Entity Relationship

Repository ? : Yes

Price : $6995 (PC)

Product Name : Normalizer

Vendor Name : Chen & Associates

Category : Data design toolkit

Primary Use : Normalize relations to 3rd Normal form

Primary User : System Analysts and Database designer

Toolkits : Data design toolkit

Methodologies : ER and relational

Repository ? : YES

Price : $1495 (PC)

Product Name : CorVison

Vendor Name : Cortex Corporation

Category : Workbench

Primary Use : Application development

Primary User : DP professionals

Toolkits : Full Workbench with code generator

Methodologies : Information Engineering

Repository ? : Yes

Price : $55,000 (PC)

Product Name : Appl. Systems Development Environment

Vendor Name : Digital Equipment Corporation

Category : Toolkit, Workbench

Primary Use : Development and maintenance during application life cycle

Primary User : Programmer/Analyst

Toolkits :

Methodologies :

Repository ? : Yes

Price :

Product Name : VM/Software Engg.

Vendor Name : IBM Corporation

Category : Methodology companion

Primary Use : Configuration Mgt. software library, tool integration

Primary User : Software engineer

Toolkits : Tool Integration capability

Methodologies : All

Repository ? : Yes

Price : $40,000

Product Name : Exelertor

Vendor Name : Index Technology

Category : Toolkit

Primary Use : Automate analysis and design

Primary User : System Analysts/Project leaders

Toolkits : Analyst

Methodologies : Gane & Sarson, Yourdon etc.

Repository ? : Yes

Price : $8,400 (PC)

Product Name : CASE*Method

Vendor Name : Oracle Corporation

Category : Methodology Companion

Primary Use : Full Systems Life Cycle, Sizing,
Generate tables, reports and forms

Primary User : Analyst, designer

Toolkits : Data flow, ER, Generator

Methodologies : CASE*Method

Repository ? : Yes

Price :

# Vita

Gautam Bose is the son of Meera and Gopal Bose. He was born in Calcutta, India. Gautam got his bachelors degree in Mechanical Engineering from Indian Institute of Technology, Madras in 1986. Before joining Lehigh University for the masters program in Computer Science, he worked for International Computers Limited (ICL) India, which is a subsidiary of ICL, United Kingdom. Gautam is currently employed with Oracle Corporation, one of the largest vendors of database and information systems software.