Lindenwood University
Digital Commons@Lindenwood University

Theses | Theses & Dissertations

1988

A Synthesized Approach for Data Base Life Cycle Design

Philip J. Morales

Follow this and additional works at: https://digitalcommons.lindenwood.edu/theses

 Part of the Business Commons

A Synthesized Approach For Data Base Life Cycle Design

by

Philip J. Morales

B. S. A. M. M.

Abstract Presented to the Faculty of
the Graduate School of Lindenwood College in
Partial Fulfillment of the Requirements for the
Degree of
MASTER of SCIENCE
1988

Committee in Charge of Candidacy:

Assistant Professor Mira Ezvan

Chairperson and Advisor

Associate Professor Robert Hulett

Senior Staff Specialist

McDonnell Douglas Missile Systems Company

Information Resource Management Mr. Tom E. Kulik

Technical Specialist

McDonnell Douglas Missile Systems Company

Information Resource Management Mr. Frank L. McArtor Jr.

DEDICATION

To Cindy and P. J. for

"Weathering the Storm."

ACKNOWLEDGEMENTS

The advice, consultation and moral support of staff and instructors given me during my tenure in graduate studies at Lindenwood College is gratefully acknowledged.

Additionally, my sincere gratitude is given to Mira Ezvan, Ph. D. who was my Faculty Advisor and of great assistance in preparing this paper.

ABSTRACT

This thesis examines the human and technological issues that are often encountered during the application-driven system design and recommends a synthesized life cycle approach to data base design. The aerospace industry is emphasized due to the highly competitive and regulatory operating environment. Special attention is given to the strategic business planning and end-user involvement in the requirements definition phase of system development. The research consists of reviewing of current literature concerning tools, techniques, and available Information Engineering (IE) methodologies that are the basis for managing data as a resource. The benefits offered by these methodologies along with the experience gained through my involvement with the implementation of a data-driven methodology within an aerospace corporation has enabled the author to develop a data base design approach called Computer Integrated Business Life Cycle methodology. The proposed approach may be applied within any manufacturing environment. However, the thesis is directed primarily toward aerospace corporations who are, or will be, involved in the transition to an integrated computer aided business environment.

Table of Contents

LIST OF ILLUSTRATIONS

Figure

I. Introduction

A. Overview. The objective of this paper is to substantiate the contention of a synthesized life cycle methodology to data base design. The synthesized approach integrates the business data requirements with data base design throughout the entire business life cycle. The business life cycle extends itself from the strategic planning stages and conceptual product design through post production support to product obsolescence. The approach supports the concept of data sharing through the use of subject data bases.

The subject data base concept recognizes a logical view of data classes by the enterprise, as opposed to the logical view offered by the traditional application-driven methodology. Subject data bases are independent of specific user applications and provide increased functionality through independent data design and storage [Martin 1982]. Some examples of subject data bases commonly used to support corporations are: Customer, Parts, Vendor, Engineering, Technical Documentation, Personnel, etc.

Implementation of a subject data base concept depends on both standardizing the definition of data classes and separating data from the application currently using the data. Data classes are logically grouped into related subject categories. Data classes resolve the enterprise's data redundancy problems and contribute to achieving data stability.

The three schema concept is also supported by the synthesized methodology. The three schema relational architecture consists of a conceptual model, external model, and an internal model. The conceptual model is the structure of the entire data base which integrates the requirements of individual users into a single "community" view.

The external model is a version of the conceptual model that can be presented to the data base management system. The internal model is the physical model that takes into consideration the distribution of data, access methods, and indexing techniques. The three schema concept provides the logical independence between data and application programs.

To best illustrate the benefits to be realized from using a life cycle approach, the focus will be placed on the aerospace industry. This is primarily because of its vast organizational structure, multi-market environment and its stringent regulatory controls in regard to information.

Furthermore, the author's experience in the implementation of a data-driven approach to structured data base design within a aerospace corporation has made possible the development of a proposed approach that avoids the pitfalls encountered frequently throughout project implementation. Additionally, the author's review of available structured data base design methodologies, including Integrated Computer

Aided Manufacturing Definition (IDEF-1X) by D. Appleton, Information Engineering (IE) by J. Martin, and 4FRONT by R. H. Holland, revealed some of the benefits to be derived from the implementation of structured methodologies. A review of these methodologies is provided.

The proposed methodology which is called Computer Integrated Business Life Cycle (CIBLC) brings about a number of positive improvements over available structured methodologies. The approach provides a basis for the development of an enterprise data model from the corporate perspective as well as user-oriented system design. It also leads to the establishment of a continuous improvement program which links business functions to corporate objectives.

While there are multiple structured data base design methodologies available, no one methodology specifically addresses the synthesized data base design approach throughout the entire business life cycle. The author is proposing an approach that can be used effectively in a manufacturing environment.

Finally, a synopsis of the proposed synthesized methodology and a discussion of management trends and current technological events will be provided.

B. Background. In the 1970s structured methodologies emerged as applications grew in complexity and greater manpower requirements

were necessary to support application development and maintenance. These structured methodologies provided an organized approach to system development through the establishment of rules, methods, and postulates which were used to organize the problem solving approach by listing, diagramming, and documenting all analytical development steps. In addition, structured methodologies assisted in the standardization and systemization of software development and maintenance approaches by applying engineering analysis techniques rather than the various approaches provided through the individual software developers.

There are three major means of classifying structured methodologies: procedure-oriented, data-oriented, and information-centered [McClure 1988].

Procedure-oriented methodology views the process as a fundamental part of the system model. These methodologies use data flow diagrams to represent the flow of data from data collection through data processing and providing the outputs. Program relationships are represented as tree structure diagrams and the detail program logic is denoted by procedural logical diagrams. These logic diagrams provide screen and report layouts to document the design of an user interface.

Hierarchical tree structure diagrams graphically represent groups of data such that the highest level (root) is made up of one group and all other groups are related to it. Procedural logic diagrams show the flow of data and relationships between individual processes.

Data-oriented methodologies view the actual data inputs and outputs as most important. Data structures are defined and the procedural components, for example "update-record" followed by "read-record" to "write-record" are derived to form the data structures. These data structures are merged to form a single hierarchical control structure and the detail procedural programming code is written to fit this structure. This approach also utilizes hierarchical tree structure diagrams and procedural logic structure diagrams.

The information- centered approach establishes a logical data model which represents the information usage throughout the entire enterprise. In the identification of system requirements, the developer analyzes the enterprise, its business objectives, and strategic information needs.

This data model is non-hierarchical and identifies all data entity types (something the enterprise wants to collect data about, i.e.. person, place or thing) and their interrelationships. The developer then builds individual information systems that handle the storage and flow of information within the organization.

Within these methodologies, two distinct schools of thought can be identified: Software Engineering and Information Engineering. The Software Engineering aspect begins with a general view of the entire

system and leads to the decomposing of each function until all sub-functions are small enough to be implemented into programming code.

In the Software Engineering approach the logical program design is separate from the physical design. Logical program design describes the data that is used within an application program and different from its actual physical form. Physical design describes the data in the form it is physically stored and is dependent on the computer hardware selected.

The Information Engineering approach supports the development of full scale systems which encompass such entities as people, procedures, data, hardware, and software that interact with each other toward achieving the enterprise goals. Information Engineering utilizes strategic planning and data modeling. Strategic planning focuses on what data is needed to accomplish the enterprise objectives while data modeling identifies the functional dependencies and associations among the entities.

Information Engineering emphasizes a logical view of data which relates to the issue of how the organization and computer systems use data. The business requirements, not the software requirements drive the system development process in the Information Engineering approach. Information Engineering can support a relational, hierarchical, or network data base design.

Present day application-driven structured methods of data base design, although functionally satisfactory, fail to meet the flexibility required to support business demands and are extremely costly. The annual budget allocated for information systems by aerospace corporations has reached levels up to $550 million in 1988 which represents 4 percent of total company revenues [ComputerWorld, 12 September, 1988].

The implementation of the application-oriented approach results in multiple data bases with duplicate data which proliferate at a rate greater than or equal to the development of applications or software packages [R. H. Holland 1988]. Application software development usually lasts longer than a year. Control over the data administration functions as well as information resource planning is absent. In addition, the approach does not consider the company's strategic initiatives as they affect the business life cycle.

Since automation constitutes approximately 12 percent of annual operating cost, it is an area where a formalized approach to company processes can substantially affect the company's financial performance [Appleton 1987].

The aerospace project in which the author participated, was initiated in response to operating in this rigid business environment. In dealing with the already established "islands of automation", the project team members concluded that further application of the application-driven approach would turn these " islands of automation"

into "continents". The approach selected by the aerospace corporation for this particular project required initial high capital investment and management commitment. Implementation of this methodology encountered many technical, political, and financial difficulties.

The methodology proposed in this thesis incorporates the necessary changes to circumvent these obstacles. The author strongly disagrees with the project's approach which is implemented from the "to-be" perspective. The "to-be" perspective identifies the data requirements and processes which are based completely on a conceptual view of the enterprise. The "to-be" perspective does not consider the already established data requirements and processes that may be unique to the enterprise.

The proposed methodology references the "as-is" environment. The "as-is" perspective reviews the present data requirements and processes with the intention of deriving the stable data requirements that can be applied to a desired future state. The data and process requirements are supplemented with new requirements as they are identified. This is based on the contention that future company processes will be driven by the same stable data of existing processes.

The critical issue facing the corporations today does not only relate to the problem of selecting a particular methodology but the need for continual search of a methodology that can fit their unique needs.

II. A Review of Available Methodologies

The following review of selected methodologies will provide a summary of advantages and disadvantages of available data base design approaches.

The Integrated Computer Aided Manufacturing Definition (IDEF-1X) approach is primarily utilized by the Department of Defense (DOD) and has been levied upon the aerospace companies competing for numerous contract awards.

The Integrated Computer Aided Manufacturing Definition (IDEF-1X) methodology offers the ability to document the conceptual schema through the use of their supporting Computer Aided Software Engineering (CASE) tool called JANUS. It provides a level of consistency necessary for integration and the sharing of data. The process activities and their relationships are extremely well documented. The methodology also establishes a matrix which identifies the various employee skills available for selection to form a well balanced project team.

In this methodology, however, strategic planning of information does not influence the system design. The system development is primarily a "bottoms-up" (application oriented) approach which is driven by user applications. The approach is process-oriented and is unable to clearly distinguish the relations between redundant processes, data origination, and similar data requirements. The methodology is limited to the analysis portion of data base design and does not address the physical design aspect.

In summary, the Integrated Computer Aided Manufacturing Definition (IDEF-1X) methodology can be extremely useful for small data base design applications and the identification of business process flows.

The Information Engineering (IE) approach was selected for review based on my personal involvement in the implementation of this approach within an aerospace corporation. The major advantage of Information Engineering is the Information Strategic Planning (ISP) stage, which enables the strategic business requirements to drive the development of supporting data base design from the data perspective of the enterprise. Information Strategic Planning (ISP) provides top management with the ability to view the enterprise in relation to goals, functions, processes, and critical success factors. The approach provides a solution to the data base design development process from the conceptual schema to logical schema and ultimately to the physical data base design.

The major disadvantage of Information Engineering is that it suggests that the development of entity-relation models should be based on each functional process area within the enterprise. This approach can be very difficult to implement due to the multitude of different entity-relation models developed for all processes within the enterprise.

Furthermore, the reconciliation of the differences between the entity-relation models is an astronomical task and nearly impossible to accomplish due to the dynamics of the models. The perspective used in

the development of the entity-relation models is also from an application process viewpoint and not from a business enterprise (global) viewpoint. These inconsistencies lead to a situation similar to attempting to hit a continuously moving target - the target being the enterprise's entity-relation model. In the environment of a large aerospace company this approach would be very costly to implement and could lead to a perpetual rework effort for data modeling.

The 4FRONT approach lends itself very closely to the Information Engineering methodology. However, the major difference between these methodologies is that the 4FRONT approach requires the up-front development of an enterprise data model and associated entity types. The enterprise data model is then provided as a target for the development of lower level individual process area data models. This approach breaks down the enterprise data requirements into manageable pieces and references a more stable target data model.

The major disadvantage of the 4FRONT approach is that it isn't widely known. Also, the supporting Computer Aided Software Engineering (CASE) tool is not as robust the Information Engineering Workbench (IEW) in the areas of flexibility, utilization of one or all of the software modules, and total support of the life cycle development phases.

The methodologies reviewed have enabled efficient processing of applications through structured application-oriented system design. However, the approaches are limited to the system design life cycle. The system design life cycle analyzes the data required to support the specific applications, develops structured programming logic, validates design with the users, implements application into a production environment and performs maintenance throughout the life of the application.

The proposed synthesized methodology expands itself beyond the limits of these approaches by the up-front integration of strategic business planning initiatives. The proposed approach also surpasses these methodologies through the identification of stable data requirements supportive of the entire business life cycle. This further emphasizes the need for the establishment of a synthesized methodology which extends over the entire enterprise.

III. Recommended Synthesized Methodology

The purpose of this chapter is to define a synthesized approach to data base life cycle design. This approach requires the total user participation and provides a method to develop systems quickly and efficiently.

The Computer Integrated Business Life Cycle (CIBLC) approach consists of four major modules: Strategic Business Information Planning (SBIP) module, Subject Area Data Analysis (SADA) module, System Design and Construction (SDC), and Maintenance. Figure (1.0) provides a structured outline of the recommended methodology.

The methodology requires establishing up-front management commitment at all levels of the organization. This is the critical success factor for obtaining the necessary resources. This can be extremely difficult in the aerospace industry because of the government cost regulations. A project of this magnitude is prohibited from charging back this effort to the customer as part of the normal operating business expenditures. The only recourse available to management is to allocate capital funding from the company's profit income.

Establishing commitment must be initiated by an executive sponsor. The sponsor should be an executive manager who adopts the proposed concept and politically nurtures it within the corporate arena. Acceptance by other executives will provide the necessary resources and establish the authority to dictate direction (as required).

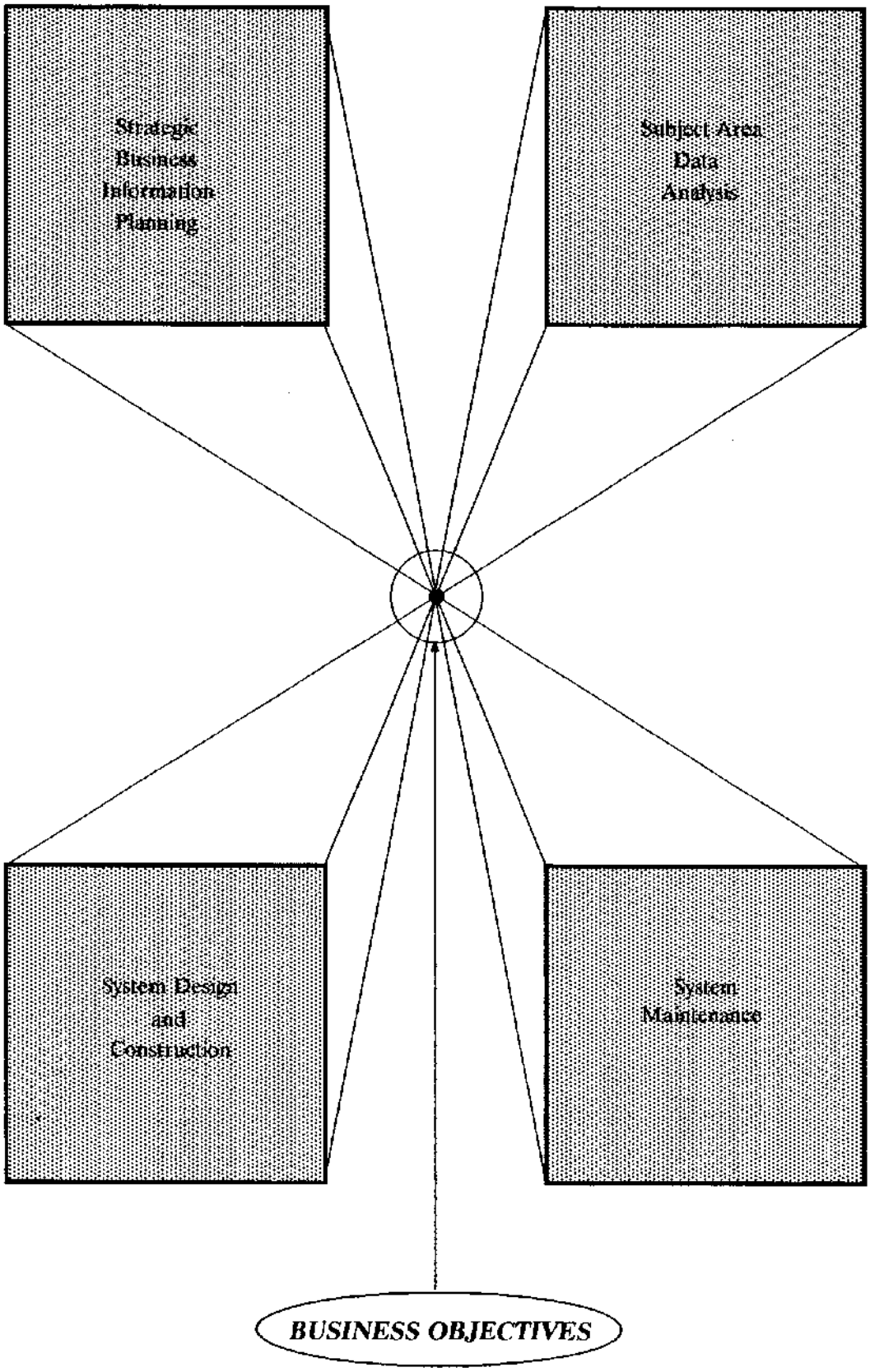


Figure 1. **Computer Integrated Business Life Cycle**

The second aspect of establishing commitment is ownership. Managers at all levels within the organization should not only have the responsibility to support this new business approach but also to participate in identifying their informational needs. This responsibility will help ease the pain of migrating to a data independent environment and will create a sense of ownership in the newly developed subject data base environment.

In addition, implementing the methodology would be extremely difficult without the support of a Computer Aided Software Engineering (CASE) tool, which would provide the flexibility and the ability to represent system specification, screen and report painters, and documentation generators with diagrams. The tool should have the ability to provide integrity checks on the resident data dictionary based on naming conventions and data definitions. The dictionary is a repository of data diagrams, data models, system specifications, and definitions. The tool should also have allowances to provide an optional forth generation programming language (i.e. Structured Query Language, SQL), code generator or the ability to interface to commercially available products.

The most important module within the Computer Integrated Business Life Cycle methodology is the Strategic Business Information Planning module. The planning module primarily enables managers to clearly review the business enterprise in terms of goals, business functions, critical success factors and the enterprise informational needs. This module establishes a major difference between the proposed methodology and the previously discussed approaches.

The Strategic Business Information Planning is composed of four areas: project management, transition planning, the enterprise data modeling, and data clustering.

The project management area provides the techniques for building a comprehensive team to gather, validate, and decide what data elements are required for functional areas. The techniques include the specific planning, implementation, and management tools for the project.

Project management depends on top management's commitment to delegating authority, providing direction and supporting the process. Management, in turn, expects the timely and cost effective achievement of subject data base development. Achieving this may require the development of an application prototype to prove the functionality of subject data bases to management. Communication to and between the functional areas is critical due to the number of different departments involved. Program management coordinates and integrates the inputs from all departments involved and facilitates the project's development by optimizing the positive effects of change to meeting the project objectives.

Three principal stages for program management are: project formation stage, implementation stage, and project completion. [Beck 1984] Project formation and direction provide the "top-down" planning which clearly defines the objectives. The implementation stage is where feedback such as status reports, budget reports, and obstacles are brought

to management's attention. Mid-course corrections may take place in order to put the project back on track. The use of automated program management planning tools such as Timeline, Task Monitor, and Quicknet may be recommended for tracking purposes.

The benefits of using automated tools will increase productivity, standardize activity reporting and provides the ability to perform "What If" analysis. These tools offer the ability to simulate cause and effect relationships for alternative comparisons.

Program management will coordinate the input of departmental critical success factors necessary to realize project cost and benefit goals. The critical success factors relate to conducting current business operations and address where the high impact areas lie along with their relative risk assessments. Establishing a measurement criteria is essential for the effective monitoring of critical success factors. These criteria should be used throughout all levels of management and should be compatible with each other. Figure (2) represents the critical success factors which support achieving a data base life cycle design.

CUSTOMER OBJECTIVES	STRATEGIC OBJECTIVES	TACTICAL OBJECTIVES
ELECTRONIC PRODUCT DATA DELIVERY.	PRODUCTION CAPABILITY FOR DIGITAL DATA TRANSFER.	IMPLEMENT NECESSARY HARDWARE/ SOFTWARE IN COMPLIANCE WITH MILITARY STANDARDS.

Figure (2) CRITICAL SUCCESS FACTORS

The team building issue must be addressed by program management because of its potential impact on schedules and the completion of the data base design from an application perspective. The concept of a Natural Working Group (NWG) for team building is recommended.

Natural Work Groups (NWG) are composed of the knowledgeable domain experts from different disciplines which naturally share the same subject data through a process, function or product within the enterprise. If necessary, these experts can discuss with other Natural Work Group (NWG) members resolving problems or improving the present process within their domains. The members of all levels of management and factory employees can be selected for natural work groups.

The NWG concept allows members to focus on identifying the data required to satisfy business needs within their respective subject areas. This ensures that data will be identified from all points of view within the entire enterprise; i. e. manufacturing, logistics, engineering. In the past, the traditional system development approach utilized interviewing techniques and questionnaires which were administered by system analysts to assess the users' data requirements. This approach may be counter productive because of the interpretation of requirements by the analyst. Implementation may be remote to the user work environment and the inefficiencies of the developed application leads to the need for rewriting of programming code.

The second area within the planning module is construction of the "as-is" software and hardware system architecture model. This model is created by constructing visual charts representing all operating system hardware, software and networks which functionally support the organization. Total team involvement is required to effectively capture and understand the organization's "as-is" architecture.

Elements of the architecture model should include all supporting hardware systems (such as mainframes, controllers, modems, workstations, etc.), all internal and external telecommunication links, and associated software. Additionally, an "as-is" functionality matrix is developed to indicate the areas where system integration exists within the enterprise. One benefit of constructing an "as-is" architecture is to provide a reference in relationship to the development of the "to-be" data base system architecture. Figure (3) is a graphical representation of an "as-is" architecture model.

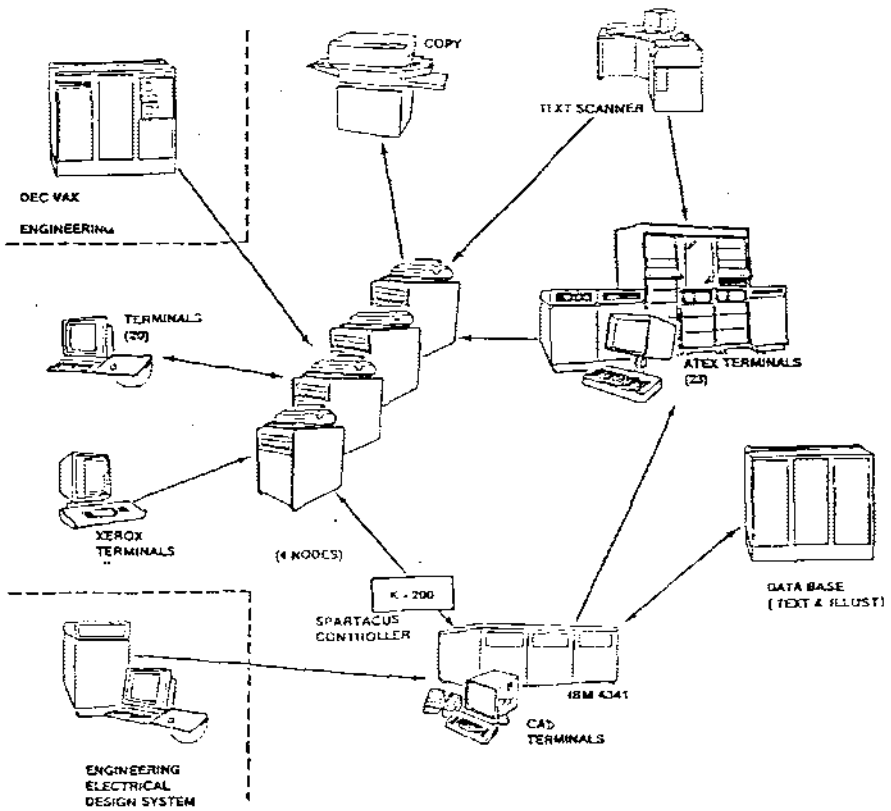


Figure (3) THE "AS-IS" SYSTEM ARCHITECTURE MODEL

Along with the "as-is" infrastructure, decomposing organizational functions down to the work processes and activities is essential. The first step in this decomposition is to provide a link between the strategic business initiatives and the computer systems requirements.

Providing this link can be accomplished through hierarchical diagrams. These diagrams illustrate relationships between the functions and processes inherent to a particular function (Reference Figure 4 and 5).

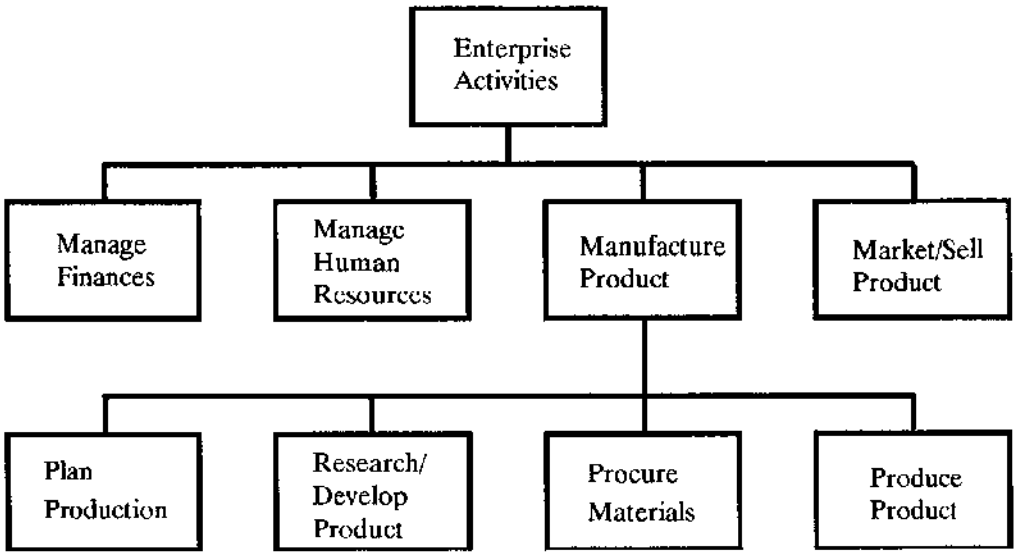


Figure (4) Functional Decomposition Diagrams

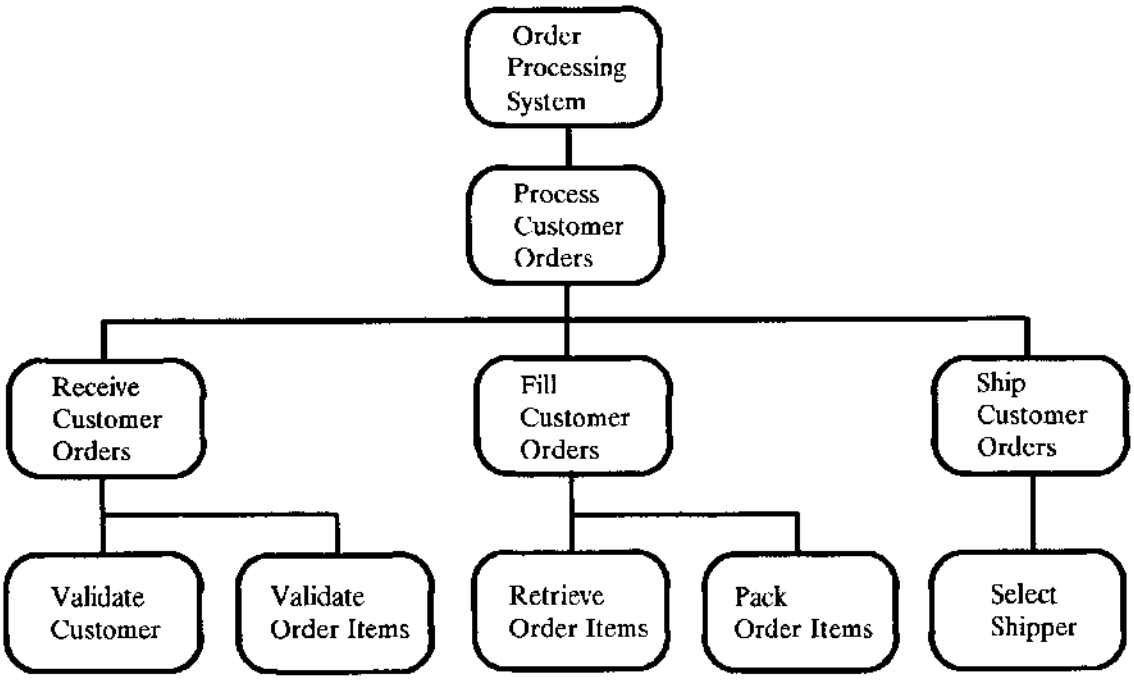


Figure (5) Process Decomposition Diagrams

A three by six diagram construction level usually provides sufficient information of major processes. [Martin 1986] The process decomposition diagrams are further decomposed into data flow diagrams. The data flow diagrams identify all internal or external data sources and requirements for each process. Associated interrelationships are established and annotated at this point.

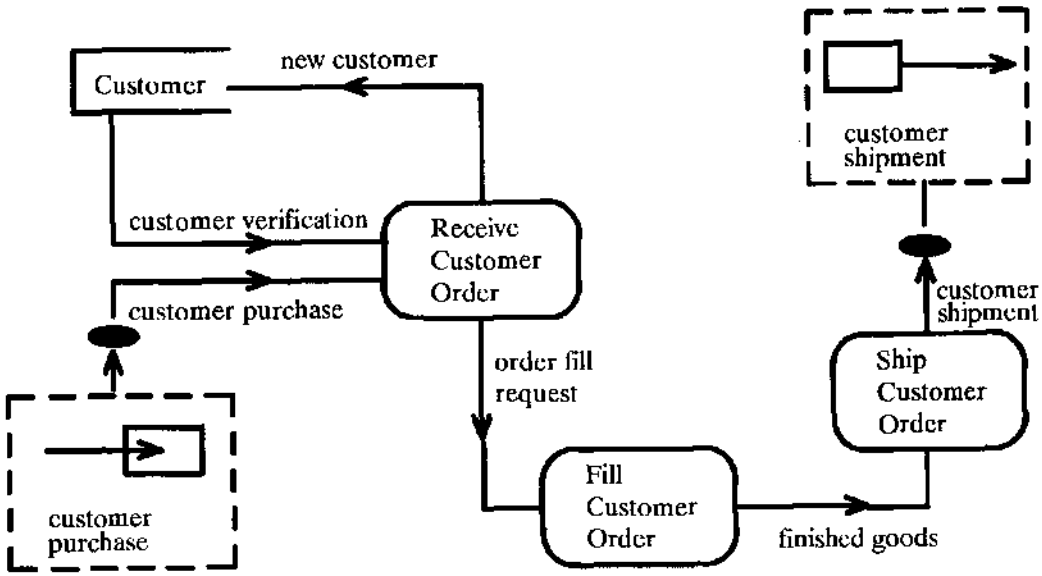


Figure (6) Data Flow Diagrams

The third step, and the major output of the planning module, is defining the "to-be" future target state of the enterprise. This output is unique to the Computer Integrated Business Life Cycle (CIBLC) approach. Defining the "to-be" state can be accomplished through assessing the potential technological impacts on the current business strategy, reviewing official corporate strategic goals, analyzing the "as-is" decomposition diagrams, assessing the economic conditions, and customer requirements. By analyzing these internal and external factors, the Natural Work Groups (NWGs) will be able to develop the "to-be" functional and process decomposition diagrams which reflect the functions and processes of the future enterprise, as shown in Figure (7).

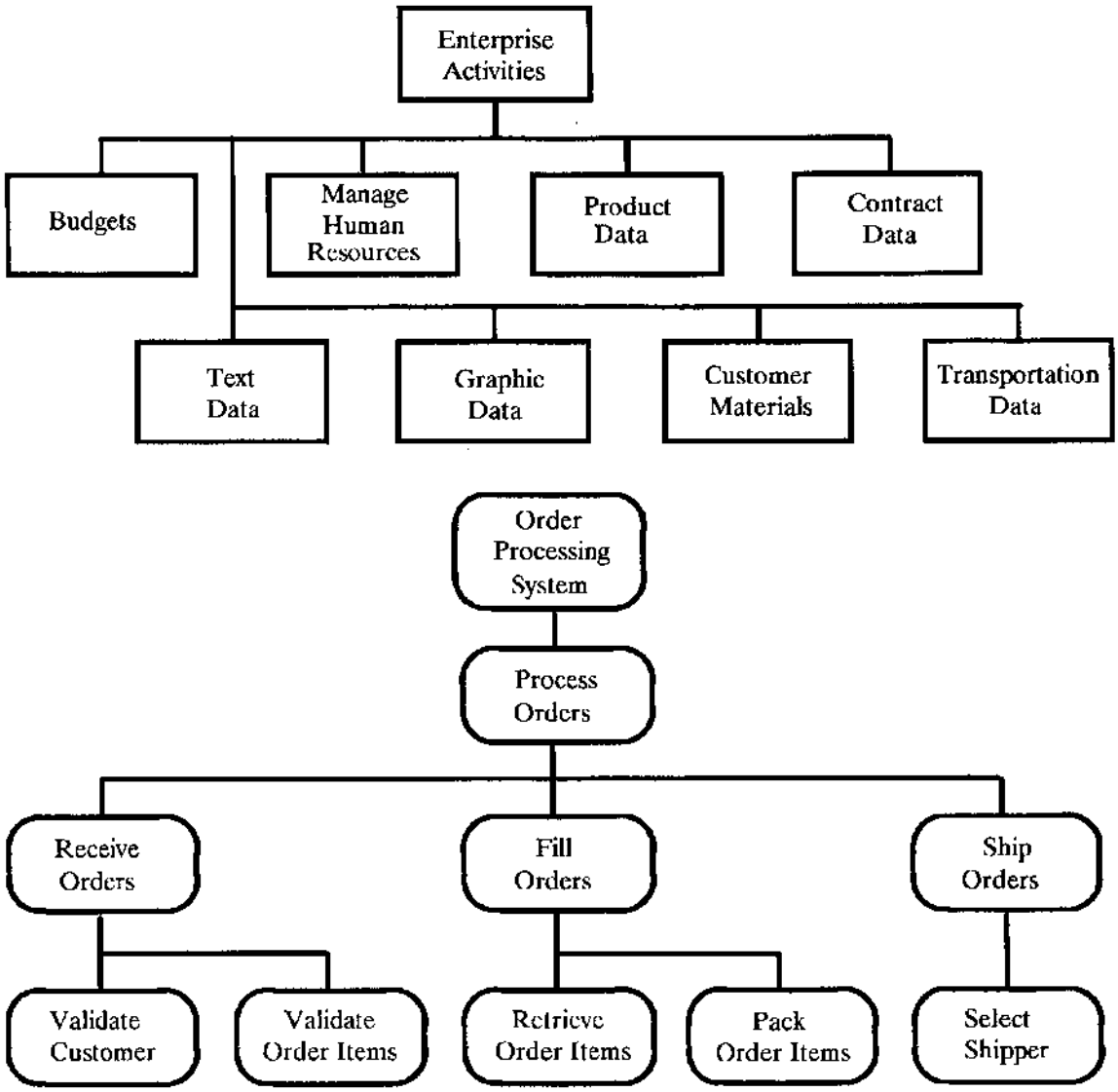


Figure (7) The "TO-BE" State Functional and Process Decomposition Diagrams

In addition, the integrating of the system life cycle phase and the product life cycle into the enterprise's overall future critical functions would reduce the time gap from initial product engineering phase to the production phase. Figure (8) and (9) represent the life cycle phases for systems development and product related processes.

Integrating the system life cycle phases into the product life cycle phases contributes to the four common support functions. They are: business management, financial management, human resource management, and property and facilities management. The business management function consists of strategic planning, operations planning, policies, practices and standards, general administration and communications. Managing finance involves financial planning, measurements and controls, accounting, payroll and benefits, and facilities accounting. Management of human resources includes human resource planning, training, labor relations, services and benefits, security and discrimination protection. The final common function is managing facilities and property, which consists of property planning, maintenance, facilities location and design, fire protection and motor pool management. Figure (10) shows the interrelationship between system life cycle and product-related life cycle for aerospace corporations.

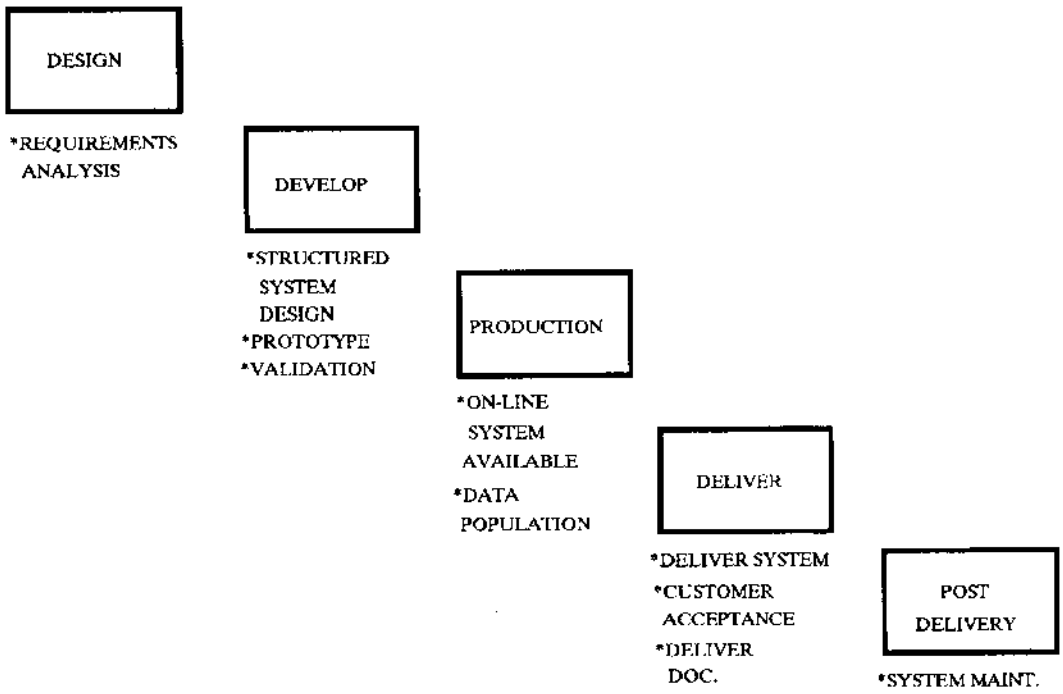


FIGURE (8) TRADITIONAL SYSTEM LIFE CYCLE PHASES

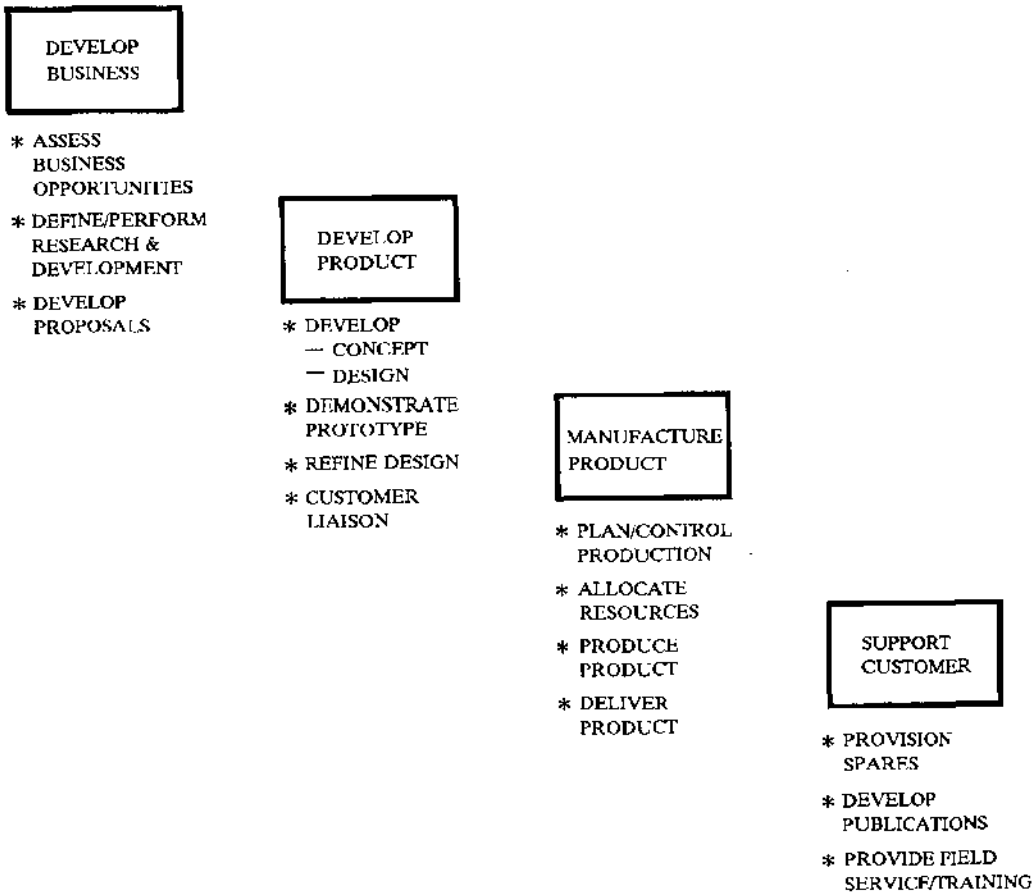


FIGURE (9) PRODUCT RELATED LIFE CYCLE FOR AEROSPACE INDUSTRIES

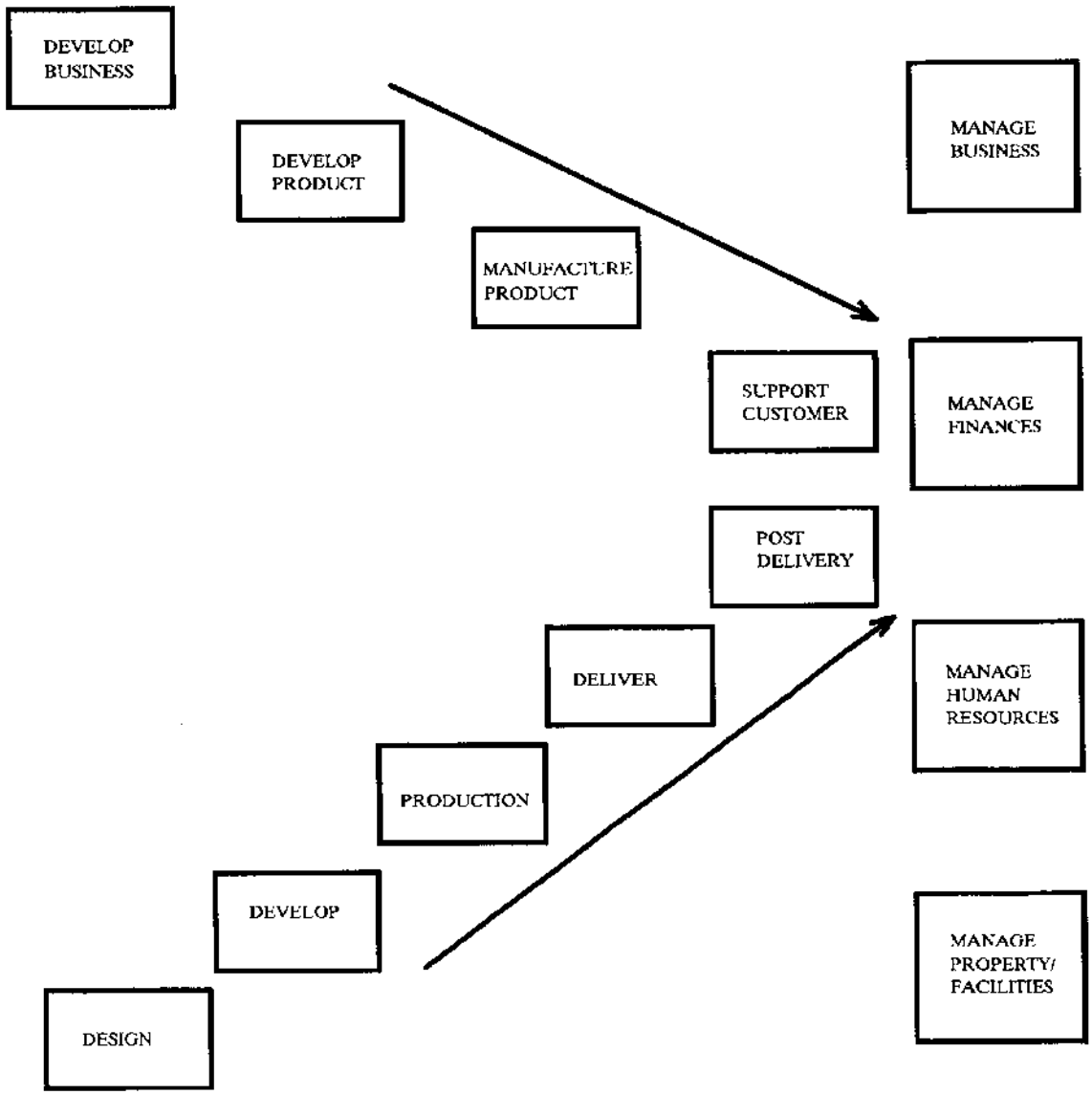


Figure (10) Integrated Support Functions

The next phase of establishing the "to-be" data model is developing the enterprise's "as-is" entity-type data pool. An entity-type data pool consists of the identification of high level entities (groupings) which represent the basic informational needs of the "to-be" enterprise. Throughout the business life cycle, these entity-types rarely change except for additions due to business consolidations, mergers, and expansions or other plans. Capturing the "as-is" entity-types provides the basic informational needs for the "to-be" target state. Caution should be exercised if plans exist concerning major corporate business expansions. which may require these informational needs to be merged with the new informational needs. Figure (11) represents "as-is" and Figure (12) represents the "to-be" entity-type data pool.

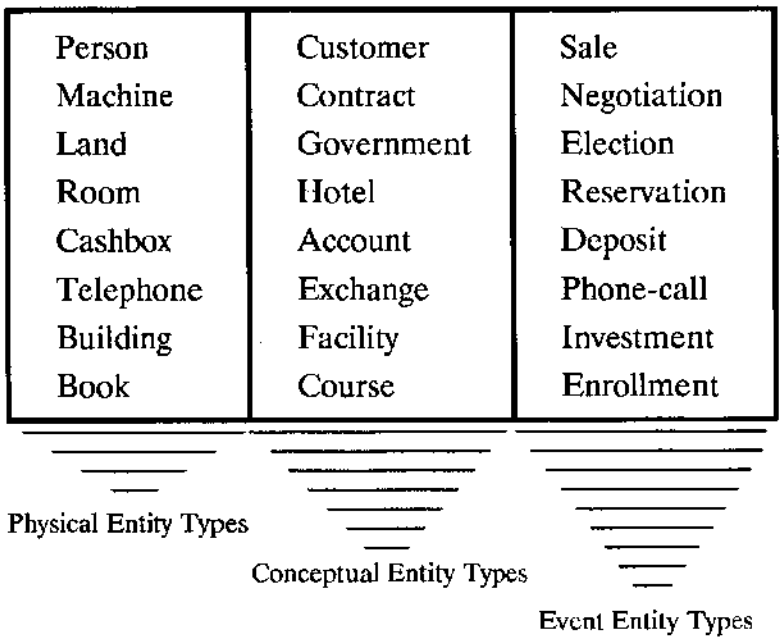


Figure (11) "AS-IS" Entity-Type Data Pool

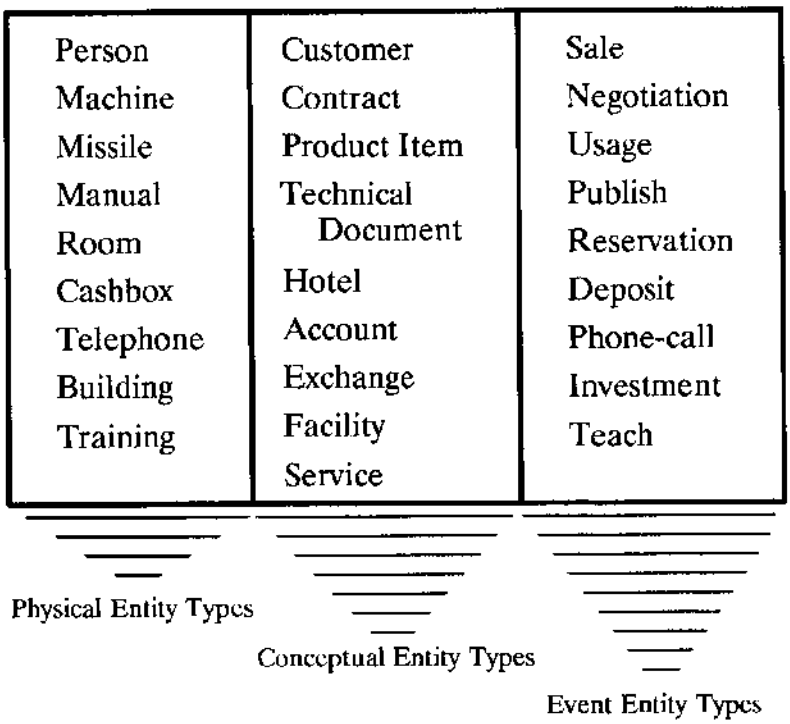


Figure (12) "TO-BE" Entity-Type Data Pool

Based on the "to-be" entity-type data pool, the Natural Work Groups (NWGs) will develop an entity-type relation model. The model is constructed using the entity-types identified and their relationships. The relationships are assigned names and are based on the enterprise's operating business rules.

Defining the "as-is" and "to-be" enterprise state enables the development of a transition plan. The purpose of the transition plan is to

smoothly migrate from the existing data base and business environment to the Strategic Business Information Planning targeted state. Upon review of the previously defined "as-is" software and hardware architecture, the critical data bases which require linking will immediately be apparent. Project management should identify the minimal level of the capital investment necessary to link the critical data bases.The further development of any new data bases and/or the enhancement of existing data bases in the old environment should not be permitted by management unless it is a contractual obligation or critical to existing business operations.

Once the enterprise entity-type relation model is constructed, further decomposition of the model is required to establish new subject areas for the enterprise business areas data requirements. The formation of subject areas can be accomplished by building various association matrixes. The matrixes define data usage based on processes, geographic location of data, and functions with supporting processes (Reference Figure 13).

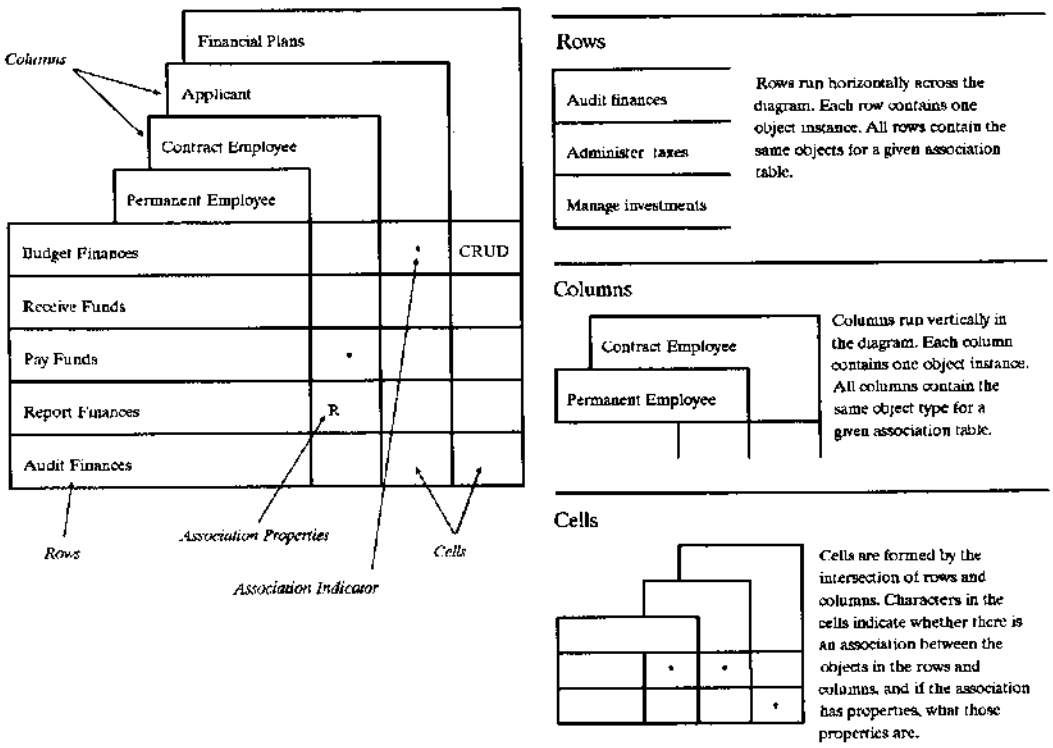


Figure (13) Association Matrixes

Association can also be established comparing data requirements and the organization that creates, reviews, updates, and deletes (CRUD charts) the data used in the particular process, as shown in Figure (14).

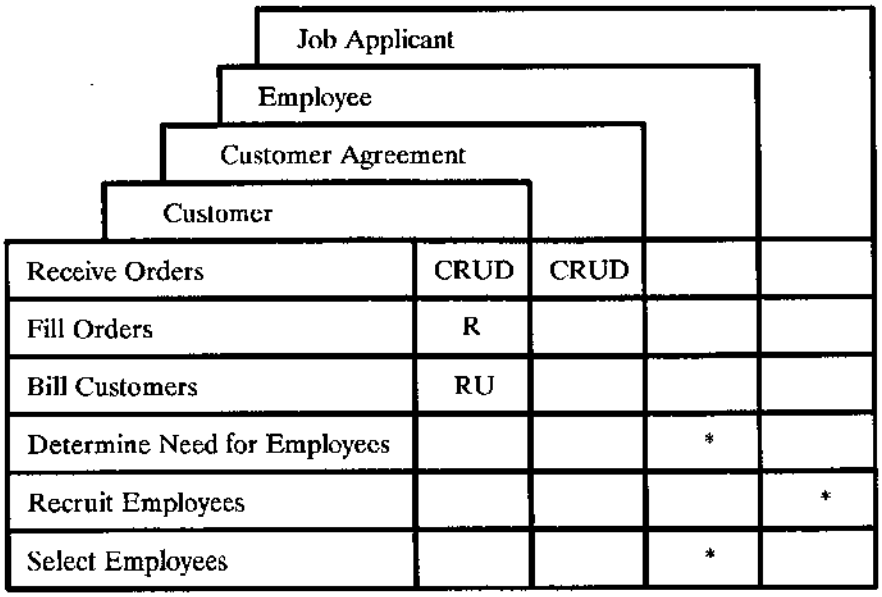


Figure (14) CRUD Charts

A data usage-to-process matrix should be established to perform affinity analysis. Affinity analysis using algorithmic formulas clusters the identified entities into subject groupings by the related processes and the selected affinity quotient defined by the analyst.

The affinity algorithm is represented by the notation [Martin 1976]:

$$E1 \text{ to } E2 = \frac{a\ (E1\ ,\ E2}{a\ (E1)}$$

E1 notation represents the number of processes using entity E1. E2 represents the number of processes using entity (E2). (E1,E2) represents the number of activities using both E1 and E2. The (a) denotes the affinity value pertaining to that entity. If (a) equals zero, then the entities should not be clustered. Otherwise, based on the developers decision on the affinity ratio desired to form a cluster, there may be as many clusters as entities. These clusters can be interpreted to form logical subject data bases. This top-down planning approach determines which business areas should be candidates for subject data bases and help prioritize their development.

When performing affinity analysis identifying ratios for data grouping is necessary. The suggested ratios are 95 percent similarity to form a grouping, 90 percent to add an object to an established grouping, 80 percent to combine groupings, and 65 percent to consider grouping of data at all. An object can be a conceptual or physical element about which data is collected and actions are performed against [Martin and Hershey 1986]. The ratios can be continuously refined based on reliability, security and other business constraints. Further analysis may be required to assign subject names to the proposed physical subject data bases. If this process is performed manually, the NWGs will be required to analyze the association matrix and groupings mathematically and then separate the data based on processes.

The second component of the Computer Integrated Business Life Cycle approach is the Subject Area Data Analysis (SADA). The data analysis component requires the development of a stable data model based on the Strategic Business Information Planning needs and the "to-be" subject area data requirements.

The first step is the development of the "to-be" entity pool for each subject area. The entities identified will be at a very detailed level and commonly used within the daily business operations. The second step is the construction of the detailed data model. Three levels of data models can exist. The entity-relations level, key-based level, and fully attributed level are shown in Figure (15).

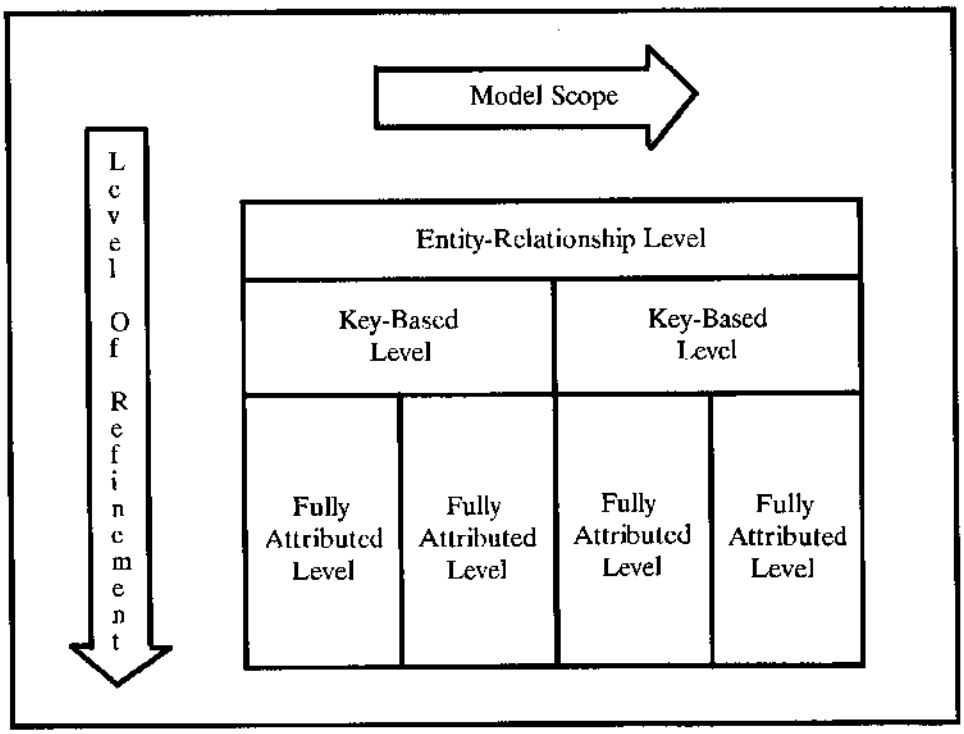
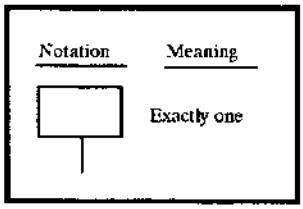


Figure (15) Levels of Entity-Relations

The entity-relation model establishes relations and cardinalities between entities. Cardinalities define numeric relationships between entities and often establish the basis for further decomposition. The primary purpose of cardinalities is not to measure the number of entities that exist, but to identify static relationships to the lowest dependent state. That is, when cardinality is represented in binary (0,1), it is at its most stable state for application development. Figure (16) shows the various notations used in defining the cardinalities of relationships between entities.



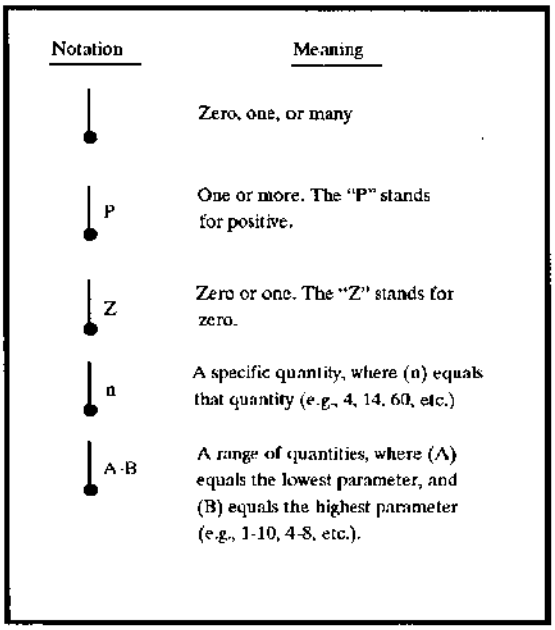


Figure (16) Cardinalities

The key-based model provides a more precise representation of the data used within the enterprise. The model primarily focuses on the identification of key attributes which uniquely describe the entity. The key-based model illustrates key migration through relationships to other entities. Migrating keys are known as "foreign keys" and provide for alternate access paths to enterprise data. A foreign key is an attribute or group of attributes within a child entity whose values exist as the primary key values in the parent entity. The key based model may represent the "as-is" or "to-be" condition. Figure (17) provides an example of a key based model.

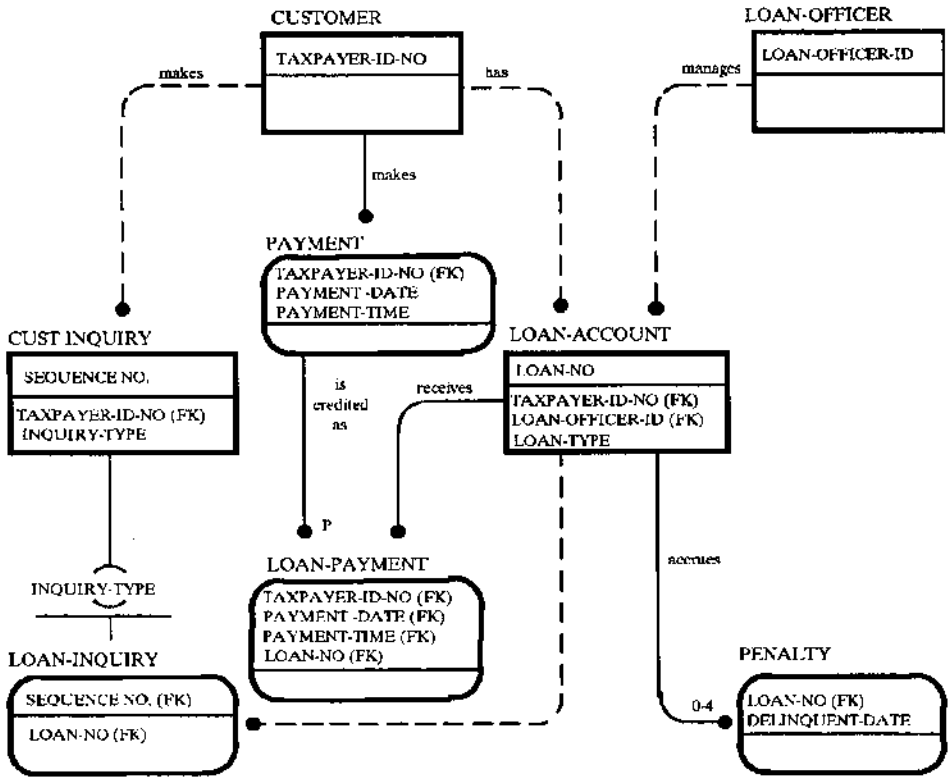


Figure (17) Key- Based Model

The fully attributed model, as illustrated in Figure (18), consists of all-key and non-key attributes and provides a stable foundation for data base design and implementation. The model also identifies all the data requirements and their interrelationships which are specific to that particular environment.

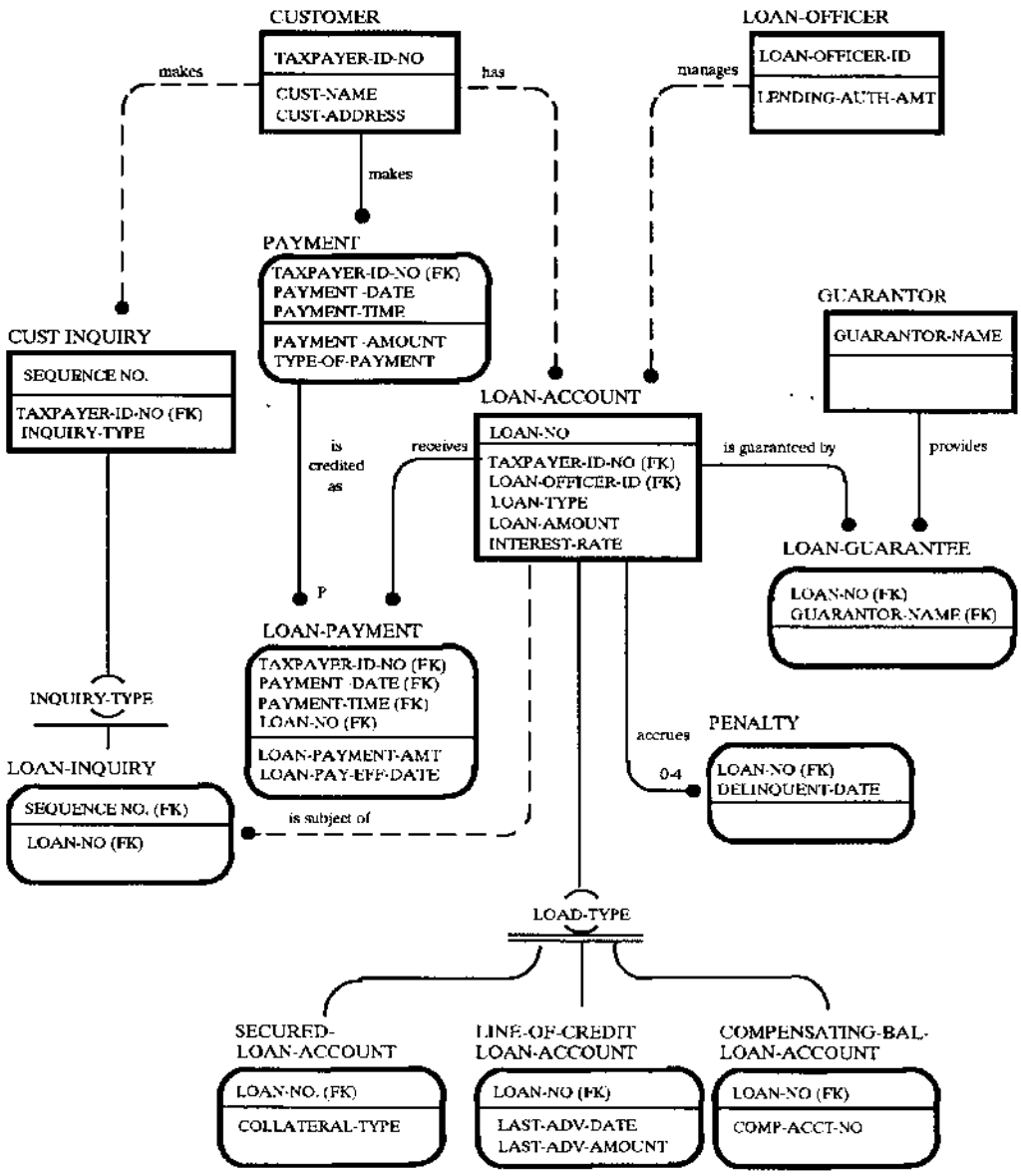


Figure (18) Fully Attributed Entity-Relation Model

The developed data model will be composed of relational tables which should be normalized. Normalization is the decomposition of compound data structures to simpler and more stable structures. The adequate and practical level of normalization is usually third normal form. Appendix A represents the rules of normalization developed by E. F. Codd, the father of relational theory [Kent 1983].

The normalized data model is analyzed to reconstruct the "to-be" processes decomposition and data flow diagrams. This analysis will provide a basic process structure for future reference and enhancements. Additionally, the NWGs will be able to perform benefit analysis by comparing the "as-is and "to-be" process models. The differences between the models can provide management with the most productive and simplified illustration of change processes supporting organizational functions. Major cost savings benefits estimated to be 50 percent for maintenance, 40 percent for reduction in redundancy and 30 percent in productivity gains are expected. [Martin 1976] The results can justify the restructuring of the organization to efficiently operate in the new "to-be" environment. Typically management is extremely sensitive and resistant to reorganization based on conceptual methodologies.

The corresponding data definitions and characteristics will be captured and maintained within the Computer Aided Software Engineering (CASE) tool.

Manual reconciliation of naming conventions, relationships between entity and/or attribute definitions may be required because of

limitations of the tool. A major benefit provided by the tool should be the capability to link subject area data models to the enterprise data model. Figure (19) represents the process flow for defining data elements.

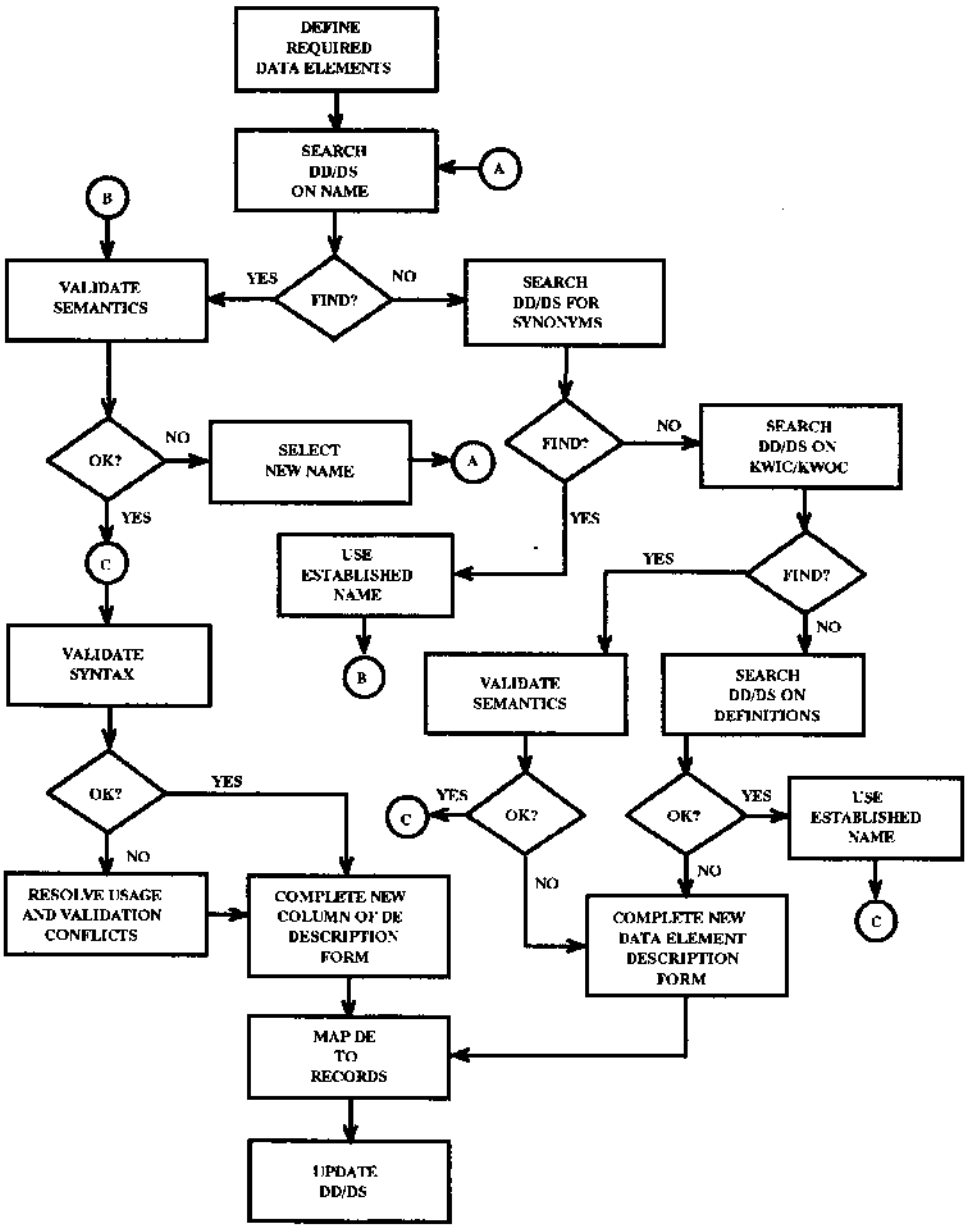


Figure (19) Naming Conventions Reconciliation Process Flow

Use of available standards will facilitate the data naming process. There may be a larger occurrence of new data names due to the concatenation of primary keys. Concatenation is a process of combining primary keys from different parent entities to form an entity that is related to both parents. Participation by all Natural Work Groups (NWGs) and the establishment of a Data Administrator must be coordinated to arrive at a standard name or acceptance of an alias which offers an appropriate level of user friendliness. Similar procedures must be followed for the reconciliation of entity relations and attributes. A need also exists for resolution between subject area data models and dictionaries as they are consolidated into one enterprise data dictionary. This consolidation can only be accomplished through close coordination with all Natural Work Groups (NWGs). In addition, the members should be required to assess the impact of the proposed changes to ensure that the data integrity will not be compromised.

The next step of the approach requires the development of an application prototype. The traditional approaches to system development often are inefficient and produce systems that do not meet the user requirements. Prototyping can be used to validate the completeness and integrity of data. The purpose of prototype development is to validate a uniform means of developing applications across all subject data bases regardless of the core process selected.

The prototype construction starts with a core process within the subject area and expands upward until it is completed. There is a need to identify all internal and external data sources which are necessary to

support the core process. This identification process which is contingent upon the prioritized sequence of selected application prototypes allows for the balanced construction of all the enterprise subject data bases. The sequence will be based on the data required to support the initial core process within the function.

The Natural Work Groups (NWGs) should clearly convey the prototype expectations to management and should establish a schedule for constructing and demonstrating the prototypes. This will reinforce sharing expectations defined to upper management.

The System Design and Construction component consist of "denormalization" of data models, creation of action diagrams, development of input/output screens, construction of physical tables and code generation. Based on performance considerations, the data model may require denormalization. Denormalization is the process of restructuring the relational tables to a lower normal form to enable efficient processing based on application design. This denormalization is a necessary part of the tuning of the data model to support the user's need for flexibility.

The Natural Work Group (NWG) members will be required to integrate the data model into the action diagrams.

The construction of action diagrams establishes the programming logic in the form of "If-Then Else" statements, as shown in Figure (20). Action diagrams also enable procedural calls and screen calls to be embedded within the diagram [Martin 1986].

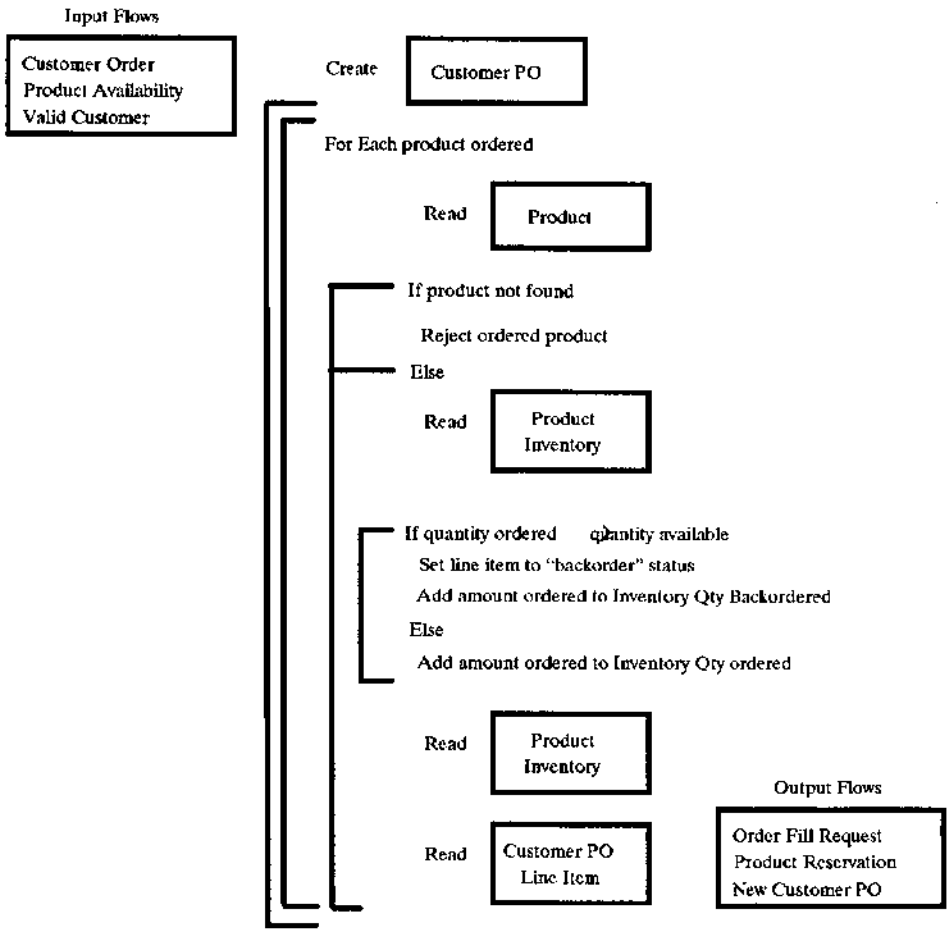


Figure (20) Action Diagram

The actual screen layouts can be developed using a screen generator. The Natural Work Groups (NWGs) can add text, format variables and graphic components anywhere on the screen. The screen layout menu should allow developers and users to see how the screen will look prior to creating the programming code.

Based on the required flexibility for subject data base design and the three schema approach, this paper will only address prototypes in a relational data base environment.

The creation of relational tables using a data definition language must be completed to support the selected prototype. In order to provide a level of portability among various relational Data Base Management Systems (DBMS), I suggest that Structured Query Language (SQL) be identified as the standard programming language for prototyping efforts be used. After the tables have been created, the emphasis should be placed on generating application programming code to access tables. In addition, selection of a more robust fourth generation language will be necessary to support more complex applications which Structured Query Language (SQL) can not support.

In order to validate the developed prototype, creating data migration translator programs is necessary. These programs provide the mechanism for moving data into the prototype environment. In addition, these programs allow batch accessing of the data during parallel processing. Parallel processing is the simultaneous or scheduled processing of applications in the new subject data base environment and the existing file structures. Although parallel processing will incur high cost, it offers a higher degree of operational stability until the existing systems are "burned off".

The next step is to populate the data base with the ported data and process the application prototype for each subject data base. At

this time, debugging the prototype may be necessary so all tables are accessed correctly. Refining the data model for performance or design changes is performed as-required. Upon completion of these changes, formal certification of the data model and prototype is affected through management participation.

The controlled transition to a production environment from the prototyping effort provides true system validation. This transition may require the formalization of a Data Resource Management (DRM) organization, similar to that shown in Figure (21).

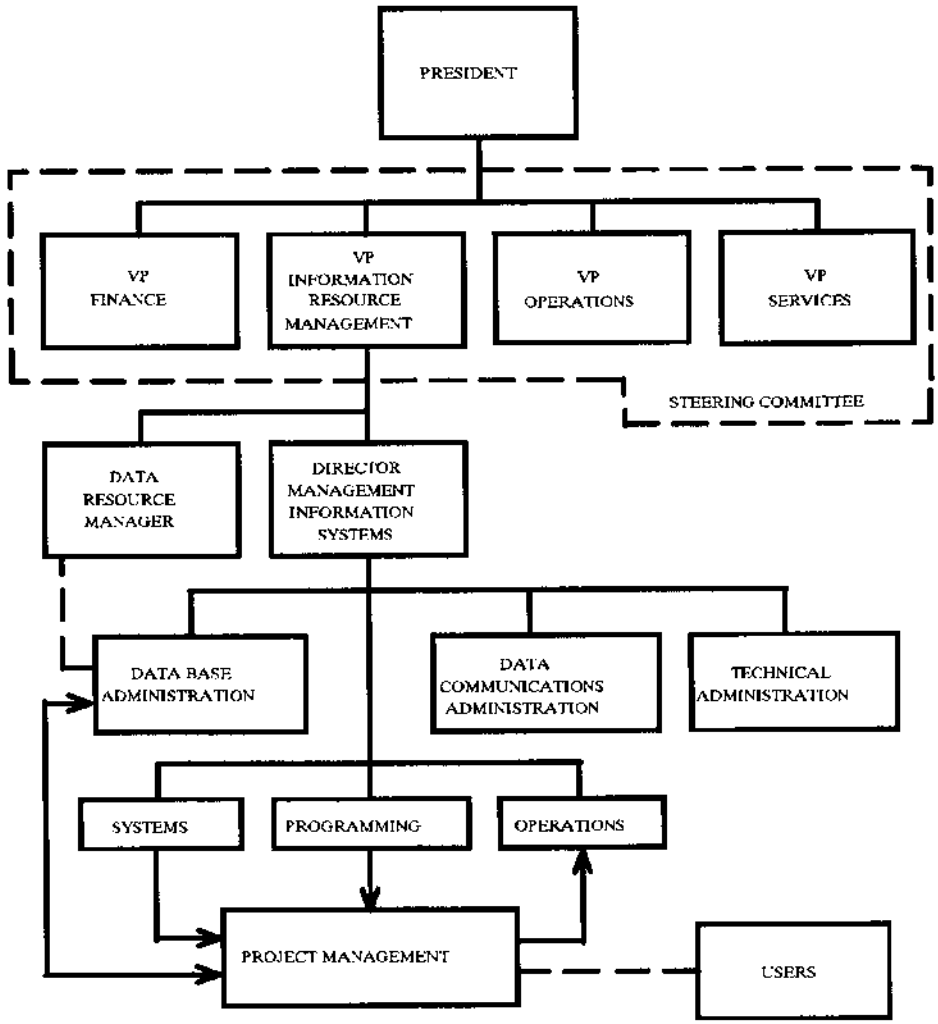


Figure (21) Data Resource Organization

Subject area data administrators are identified for each subject area. The administrators support several aerospace projects within subject areas. In this situation the administrator may spend the majority of his time arbitrating between project and functional data definitions and structures.

Utilizing the translator programs, the existing data should be ported into the production subject data base environment. The testing and validation phase must be performed to ensure data integrity, data accuracy, security, user access, query capability, and performance before global access is granted. This stage also requires the delivery of system documentation and tutorials. Maintenance of these documents is required for the life of the system and may be in automated form.

The final and never ending component of the Computer Integrated Business Life Cycle approach is system maintenance. This can be accomplished by the implementation of the Continuous Improvement Program (CIP) (inferred from the Japanese management approach called KAIZEN) which is unique to the Computer Integrated Business Life Cycle approach. This approach supports the Natural Work Group (NWG) concept and close employee and management relations. In the aerospace industry, customers, subcontractors, and vendors are also included in striving for continuous improvement. The approach will also be a major input into the Strategic Business Information Planning process and dictate the necessary changes to the process decomposition diagrams in meeting future business demands. Constant resolution of

data dictionary between subject areas will also be performed to ensure data stability throughout the enterprise.

Through the implementation of the Computer Integrated Business Life Cycle methodology, the enterprise evolves from today's business environment to a more cost effective, competitive, and flexible business environment of the 1990's.

IV. Conclusion

The previous discussions has alleged a problem of information mismanagement within the aerospace industry and have recommended a synthesized methodological approach for subject data base development. Additionally, key data and management issues associated with the traditional application-oriented system design and integrated business system life cycle approach were discussed.

Traditional application-oriented data base design life cycle approaches start with a requirements analysis, followed by structured application design, design validation, on-line production capability, customer acceptance and maintenance. The most important common characteristic among such system life cycles is that they are only concerned with software. The critical data requirements analysis is performed only to define the data needs required by the software, not the enterprise's informational needs. Additionally, in the traditional system design cycle all steps are performed in a linear manner. The scope of each step is specific and does not take advantage of the benefits from similar techniques.

There are two main differences between the proposed Computer Integrated Business Life Cycle approach and the methodologies reviewed. First, the proposed approach is data-oriented. The data requirements analysis is performed on an enterprise scale, beginning with strategic planning together with corporate policies and product data requirements. Secondly, the proposed methodology synthesizes the reviewed methodologies into a cohesive approach for an integrated subject data base environment.

Each module of the Computer Integrated Business Life Cycle approach strongly addresses areas were other methodologies are weak. The Strategic Business Information Planning module establishes the critical mapping between corporate goals and the information needs. The Subject Area Data Analysis module defines the enterprise data model and further partitions the enterprise into subject areas proving increased accuracy of data requirements. These two modules establish the critical differences in relation to the methodologies reviewed. The System Design and Construction modules, through the use of action diagrams provide the essential consistency between the conceptual data model and the physical design. The Maintenance module in conjunction with the Continuous Improvement Program provides the necessary refinement of the enterprise model to support changing business requirements.

The recommended Computer Integrated Business Life Cycle Business approach provides the additional capability beyond the traditional system development approach and the business life cycle data needs. The intention of the proposed methodology is to integrate the necessary business functions to ensure continuous improvement for the life of the enterprise. The synthesized approach allows tailoring of the methodology to take into account the idiosyncrasies of the business environment.

The proposed methodology along with available Computer Aided Software Engineering (CASE) tools, and structured analysis techniques

are only a partial solution to managing data as a resource. Management must begin to understand the technological and personnel issues that are inevitable as the transition occurs to an integrated data base environment. The critical factor to future success is the level of management commitment. If management is not willing to pay the price of integration today, there will be no tomorrow for that enterprise.

APPENDIX A

COMPUTING PRACTICES

A SIMPLE GUIDE TO FIVE NORMAL FORMS IN RELATIONAL DATABASE THEORY

WILLIAM KENT *International Business Machines Corporation*

SUMMARY: The concepts behind the five principal normal forms in relational database theory are presented in simple terms.

1. INTRODUCTION

The normal forms defined in relational database theory represent guidelines for record design. The guidelines corresponding to first through fifth normal forms are presented, in terms that do not require an understanding of relational theory. The design guidelines are meaningful even if a relational database system is not used. We present the guidelines without referring to the concepts of the relational model in order to emphasize their generality and to make them easier to understand. Our presentation conveys an intuitive sense of the intended constraints on record design, although in its informality it may be imprecise in some technical details. A comprehensive treatment of the subject is provided by Date [4].

The normalization rules are designed to prevent update anomalies and data inconsistencies. With respect to performance trade-offs, these guidelines are biased toward the assumption that all nonkey fields will be updated frequently. They tend to penalize retrieval, since data which may have been retrievable from one record in an unnormalized design may have to be retrieved from several records in the normalized form. There is no obligation to fully normalize all records when actual performance requirements are taken into account.

2. FIRST NORMAL FORM

First normal form [1] deals with the "shape" of a record type. Under first normal form, *all occurrences of a record type must contain the same number of fields*. First normal form excludes variable repeating fields and groups. This is not so much a design guideline as a matter of definition. Relational database theory does not deal with records having a variable number of fields.

3. SECOND AND THIRD NORMAL FORMS

Second and third normal forms [2, 3, 7] deal with the relationship between nonkey and key fields. Under second and third normal forms, *a nonkey field must provide a fact about the key, the whole key, and nothing but the*

Author's Present Address: William Kent, International Business Machines Corporation, General Products Division, Santa Teresa Laboratory, San Jose, CA

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.
© 1983 ACM 0001-0782/83/0200-0120 75c

In addition, the record must satisfy first normal form.

We deal now only with "single-valued" facts. A single-valued fact could be a one-to-many relationship such as the department of an employee or a one-to-one relationship such as the spouse of an employee. Thus, the phrase "Y is a *fact* about X" signifies a one-to-one or one-to-many relationship between Y and X. In the general case, Y might consist of one or more fields and so might X. In the following example, QUANTITY is a fact about the combination of PART and WAREHOUSE.

3.1 Second Normal Form

Second normal form is violated when a nonkey field is a fact about a subset of a key. It is only relevant when the key is composite, i.e., consists of several fields. Consider the following inventory record.

PART	WAREHOUSE	QUANTITY	WAREHOUSE-ADDRESS
key	key		

The key here consists of the PART and WAREHOUSE fields together, but WAREHOUSE-ADDRESS is a fact about the WAREHOUSE alone. The basic problems with this design are:

- The warehouse address is repeated in every record that refers to a part stored in that warehouse.
- If the address of the warehouse changes, every record referring to a part stored in that warehouse must be updated.
- Because of the redundancy, the data might become inconsistent, with different records showing different addresses for the same warehouse.
- If at some point in time there are no parts stored in the warehouse, there may be no record in which to keep the warehouse's address.

To satisfy second normal form, the record shown above should be decomposed into (replaced by) the two records:

PART	WAREHOUSE	QUANTITY
key	key	

WAREHOUSE	WAREHOUSE-ADDRESS
key	

When a data design is changed in this way, i.e., replacing unnormalized records with normalized records, the process is referred to as *normalization*. The term "normalization" is sometimes used relative to a particular normal form. Thus, a set of records may be normalized with respect to second normal form but not with respect to third.

The normalized design enhances the integrity of the data by minimizing redundancy and inconsistency, but at some possible performance cost for certain retrieval applications. Consider an application that wants the addresses of all warehouses stocking a certain part. In the unnormalized form, the application searches one record type. With the normalized design, the application has to search two record types and connect the appropriate pairs.

3.2 Third Normal Form

Third normal form is violated when a nonkey field is a fact about another nonkey field, as in

EMPLOYEE	DEPARTMENT	LOCATION
key		

The EMPLOYEE field is the key. If each department is located in one place, then the LOCATION field is a fact about the DEPARTMENT—in addition to being a fact about the EMPLOYEE. The problems with this design are the same as those caused by violations of second normal form.

- The department's location is repeated in the record of every employee assigned to that department.
- If the location of the department changes, every such record must be updated.
- Because of the redundancy, the data might become inconsistent, e.g., different records showing different locations for the same department.
- If a department has no employees, there may be no record in which to keep the department's location.

To satisfy third normal form, the record shown above should be decomposed into the two records:

EMPLOYEE	DEPARTMENT
key	

DEPARTMENT	LOCATION
key	

To summarize, a record is in second and third normal forms if every field is either part of the key or provides a (single-valued) fact about exactly the whole key and nothing else.

3.3 Functional Dependencies

In relational database theory, second and third normal forms are defined in terms of *functional dependencies*, which correspond approximately to our single-valued facts. A field Y is "functionally dependent" on a field (or fields) X if it is invalid to have two records with the same X value but different Y values. That is, a given X value must always occur with the same Y value. When X is a key, then all fields are by definition functionally dependent on X in a trivial way, since there cannot be two records having the same X value.

There is a slight technical difference between functional dependencies and single-valued facts as we have presented them. Functional dependencies only exist when the things involved have unique and singular identifiers (representations). For example, suppose a person's address is a single-valued fact, i.e., a person has only one address. If we do not provide unique identifiers for people, then there will not be a functional dependency in the data.

PERSON	ADDRESS
John Smith	123 Main St., New York
John Smith	321 Center St., San Francisco

Although each person has a unique address, a given *name* can appear with several different addresses. Hence, we do not have a functional dependency corresponding to our single-valued fact. Similarly, the address has to be spelled identically in each occurrence in order to have a functional dependency. In the following case, the same person appears to be living at two different addresses, again precluding a functional dependency.

PERSON	ADDRESS
John Smith	123 Main St., New York
John Smith	123 Main Street, NYC

We are not defending the use of nonunique or nonsingular representations. Such practices often lead to data maintenance problems of their own. We do wish to point out, however, that functional dependencies and the various normal forms are really only defined for situations in which there are unique and singular identifiers. Thus, the design guidelines as we present them are a bit stronger than those implied by the formal definitions of the normal forms.

For instance, we as designers know that in the following example there is a single-valued fact about a nonkey field, and hence the design is susceptible to all the update anomalies mentioned earlier.

EMPLOYEE	FATHER	FATHER'S-ADDRESS
Art Smith	John Smith	123 Main St., New York
Bob Smith	John Smith	123 Main Street, NYC
Cal Smith	John Smith	321 Center St., San Francisco

However, in formal terms, there is no functional dependency here between FATHER'S-ADDRESS and FATHER, and hence, no violation of third normal form.

4. FOURTH AND FIFTH NORMAL FORMS

Fourth [5] and fifth [6] normal forms deal with multivalued facts. A multivalued fact may correspond to a many-to-many relationship, as with employees and skills, or to a many-to-one relationship, as with the children of an employee (assuming only one parent is an employee). By "many-to-many" we mean that an employee may have several skills and/or a skill may belong to several employees. Note that we look at the many-to-one relationship between children and fathers as a single-valued fact about a child but a multivalued fact about a father.

In a sense, fourth and fifth normal forms are also about composite keys. These normal forms attempt to minimize the number of fields involved in a composite key, as suggested by the examples that follow.

4.1 Fourth Normal Form

Under fourth normal form, *a record type should not contain two or more independent multivalued facts about an entity.* In addition, the record must satisfy third normal form. The term "independent" will be discussed after considering an example.

Consider employees, skills, and languages, where an employee may have several skills and several languages. We have here two many-to-many relationships, one between employees and skills, and one between employees and languages. Under fourth normal form, these two relationships should not be represented in a single record such as

EMPLOYEE	SKILL	LANGUAGE
	key	

Instead, they should be represented in the two records

EMPLOYEE	SKILL
	key

EMPLOYEE	LANGUAGE
	key

Note that other fields, not involving multivalued facts, are permitted to occur in the record, as in the case of the QUANTITY field in the earlier PART/WAREHOUSE example.

The main problem with violating fourth normal form is that it leads to uncertainties in the maintenance policies. Several policies are possible for maintaining two independent multivalued facts in one record.

(1) A disjoint format, in which a record contains either a skill or a language, but not both.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	
Smith	type	
Smith		French
Smith		German
Smith		Greek

This is not much different from maintaining two separate record types. We note in passing that such a format also leads to ambiguities regarding the meanings of blank fields. A blank SKILL could mean the person has no skill, that the field is not applicable to this employee, that the data is unknown, or, as in this case, that the data may be found in another record.

(2) A random mix, with three variations

(a) Minimal number of records with repetitions.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	French
Smith	type	German
Smith	type	Greek

(b) Minimal number of records, with null values.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	French
Smith	type	German
Smith		Greek

(c) Unrestricted.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	French
Smith	type	
Smith		German
Smith	type	Greek

(3) A "cross-product" form where, for each employee there must be a record for every possible pairing of one of his skills with one of his languages.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	French
Smith	cook	German
Smith	cook	Greek
Smith	type	French
Smith	type	German
Smith	type	Greek

her problems caused by violating fourth normal form
similar in spirit to those mentioned earlier for violaons of second or third normal form. They take different
nations depending on the chosen maintenance policy.

f there are repetitions, then updates have to be done in
ltiple records, and the records could become inconsisnt.
Insertion of a new skill may involve looking for a record
th a blank skill, inserting a new record with a possibly
nk language, or inserting multiple records pairing the
w skill with some or all of the languages.
Deletion of a skill may involve blanking out the skill
d in one or more records (perhaps with a check that
s does not leave two records with the same language
a blank skill) or deleting one or more records, coupled
h a check that the last mention of some language has
been deleted also.

urth normal form minimizes such update problems.

4.1 *Independence* We mentioned *independent* multi-
ued facts earlier, and we now illustrate what we mean
that term. The two many-to-many relationships, emoyee:skill and employee:language, are independent in
t there is no direct connection between skills and lanages. There is only an indirect connection because they
done to some common employee. That is, it does not
tter which skill is paired with which language in a
cord: the pairing does not convey any information. That
precisely why all the maintenance policies mentioned
ier can be allowed.
In contrast, suppose that an employee can only exercise
rtain skills in certain languages. Perhaps Smith can
ok French cuisine only, but can type French, German,
d Greek. Then the pairing of skills and languages bemes meaningful, and there is no longer an ambiguity of
intenance policies. In the present case, only the follow- form is correct.

EMPLOYEE	SKILL	LANGUAGE
Smith	cook	French
Smith	type	French
Smith	type	German
Smith	type	Greek

Thus, the employee:skill and employee:language relanships are no longer independent. These records do not
olate fourth normal form. When there is an interdeendence among the relationships, it is acceptable to repent them in a single record.

4.2. *Multivalued Dependencies* For readers interested
pursuing the technical background of fourth normal
m a bit further, we mention that fourth normal form is defined in terms of *multivalued dependencies* that correspond to our independent multivalued facts. Multivalued dependencies, in turn, are defined essentially as relationships that accept the "cross-product" maintenance policy mentioned above. For our example, every one of an employee's skills must appear paired with every one of his languages. It may or may not be obvious to the reader that this is equivalent to our notion of independence; since every possible pairing must be present, there is no "information" in the pairings. Such pairings convey information only if some of them can be absent, i.e., only if it is possible that some employee cannot perform some skill in some language. If all pairings are always present, then the relationships are really independent.

We should also point out that multivalued dependencies and fourth normal form also apply to relationships involving more than two fields. For example, suppose we extend the earlier example to include projects, in the following sense:

- An employee uses certain skills on certain projects.
- An employee uses certain languages on certain projects.

If there is no direct connection between the skills and languages that an employee uses on a project, then we could treat this as two independent many-to-many relationships of the form EP:S and EP:L, where EP represents a combination of an employee with a project. A record including employee, project, skill, and language would violate fourth normal form. Two records, containing fields E, P, S and E, P, L, respectively, would satisfy fourth normal form.

4.2. Fifth Normal Form

**Fifth normal form deals with cases where information can be reconstructed from smaller pieces of information which can be maintained with less redundancy. Second, third, and fourth normal forms also serve this purpose, but fifth normal form generalizes to cases not covered by the others.

We will not attempt a comprehensive exposition of fifth normal form, but will illustrate the central concept with a commonly used example, namely, one involving agents, companies, and products. If agents represent companies, companies make products, and agents sell products, then we might want to keep a record of which agent sells which product for which company. This information could be kept in one record type with three fields:

AGENT	COMPANY	PRODUCT
Smith	Ford	car
Smith	GM	truck

This form is necessary in the general case. For example, although agent Smith sells cars made by Ford and trucks made by GM, he does not sell Ford trucks or GM cars. Thus, we need the combination of all three fields to know which combinations are valid and which are not.

But suppose that a certain rule is in effect: if an agent sells a certain product and he represents the company making that product, then he sells that product for that company.

**Reference cases are in the previous second, third and fourth normal forms.

AGENT	COMPANY	PRODUCT
Smith	Ford	car
Smith	Ford	truck
Smith	GM	car
Smith	GM	truck
Jones	Ford	car

In this case, it turns out that we can reconstruct all the true facts from a normalized form consisting of three separate record types, each containing two fields.

AGENT	COMPANY
Smith	Ford
Smith	GM
Jones	Ford

AGENT	PRODUCT
Smith	car
Smith	truck
Jones	car

COMPANY	PRODUCT
Ford	car
Ford	truck
GM	car
GM	truck

These three record types are in fifth normal form, whereas the corresponding three-field record shown previously is not.

Roughly speaking, we may say that a record type is in fifth normal form when its information content cannot be reconstructed from several smaller record types, i.e., from record types each having fewer fields than the original record. The case where all the smaller records have the same key is excluded. If a record type can only be decomposed into smaller records which all have the same key, then the record type is considered to be in fifth normal form without decomposition. A record type in fifth normal form is also in fourth, third, second, and first normal forms.

Fifth normal form does not differ from fourth normal form unless there exists a symmetric constraint such as the rule about agents, companies, and products. In the absence of such a constraint, a record type in fourth normal form is always in fifth normal form.

One advantage of fifth normal form is that certain redundancies can be eliminated. In the normalized form, the fact that Smith sells cars is recorded only once; in the unnormalized form, it may be repeated many times.

It should be observed that although the normalized form involves more record types, there may be fewer total record occurrences. This is not apparent when there are only a few facts to record, as in the example shown above. The advantage is realized as more facts are recorded, since the size of the normalized files increases in an additive fashion, while the size of the unnormalized file increases in a multiplicative fashion. For example, if we add a new agent who sells x products for y companies, where each of these companies makes each of these products, we have to add $x + y$ new records to the normalized form, but $x \cdot y$ new records to the unnormalized form.

It should also be noted that all three record types are required in the normalized form in order to reconstruct the same information. From the first two record types shown above we learn that Jones represents Ford and that Ford makes trucks. But we cannot determine whether Jones sells Ford trucks until we look at the third record type to determine whether Jones sells trucks at all.

The following example illustrates a case in which a rule about agents, companies, and products is satisfied and which clearly requires all three record types in the normalized form. Any two of the record types taken alone will imply something untrue.

AGENT	COMPANY	PRODUCT
Smith	Ford	car
Smith	Ford	truck
Smith	GM	car
Smith	GM	truck
Jones	Ford	car
Jones	Ford	truck
Brown	Ford	car
Brown	GM	car
Brown	Tovota	car
Brown	Tovota	bus

AGENT	COMPANY
Smith	Ford
Smith	GM
Jones	Ford
Brown	Ford
Brown	GM
Brown	Tovota

Fifth Normal Form

COMPANY	PRODUCT
Ford	car
Ford	truck
GM	car
GM	truck
Tovota	car
Tovota	bus

Fifth Normal Form

AGENT	PRODUCT
Smith	car
Smith	truck
Jones	car
Jones	truck
Brown	car
Brown	bus

Fifth Normal Form

Observe that:

- Jones sells cars and GM makes cars, but Jones does not represent GM.
- Brown represents Ford and Ford makes trucks, but Brown does not sell trucks.
- Brown represents Ford and Brown sells buses, but Ford does not make buses.

Fourth and fifth normal forms both deal with combinations of multivalued facts. One difference is that the facts dealt with under fifth normal form are not independent, in the sense discussed earlier. Another difference is that, although fourth normal form can deal with more than two multivalued facts, it only recognizes them in pairwise groups. We can best explain this in terms of the normalization process implied by fourth normal form. If a record violates fourth normal form, the associated normalization process decomposes it into two records, each

ontaining fewer fields than the original record. Any of hese violating fourth normal form is again decomposed nto two records, and so on until the resulting records are ll in fourth normal form. At each stage, the set of records fter decomposition contains exactly the same informaion as the set of records before decomposition.

In the present example, no pairwise decomposition is ossible. There is no combination of two smaller records hich contains the same total information as the original ecord. All three of the smaller records are needed. ence, an information-preserving pairwise decomposition s not possible, and the original record is not in violation f fourth normal form. Fifth normal form is needed in rder to deal with the redundancies in this case.

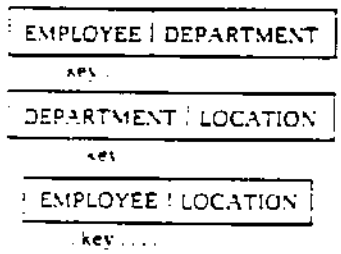
UNAVOIDABLE REDUNDANCIES

ormalization certainly does not remove all redundanes. Certain redundancies seem to be unavoidable, particiarly when several multivalued facts are dependent ather than independent. In the example shown in Sec. 1.1, it seems unavoidable that we record the fact that Smith can type" several times. Also, when the rule about ents, companies, and products is not in effect, it seems navoidable that we record the fact that "Smith sells rs" several times.

INTERRECORD REDUNDANCY

he normal forms discussed here deal only with redunancies occurring within a single record type. Fifth norhal form is considered to be the ultimate normal form ith respect to such redundancies [6].

Other redundancies can occur across multiple record pes. For the example concerning employees, departents, and locations, the following records are in third ormal form in spite of the obvious redundancy:



fact, two copies of the same record type would constite the ultimate in this kind of undetected redundancy. terrecord redundancy has been recognized for some me [1], and has recently been addressed in terms of rmal forms and normalization [8].

7. CONCLUSION

While we have tried to present the normal forms in a simple and understandable way, we are by no means suggesting that the data design process is correspondingly simple. The design process involves many complexities that are quite beyond the scope of this paper. In the first place, an initial set of data elements and records has to be developed as candidates for normalization. Then the factors affecting normalization have to be assessed:

- Single-valued vs. multivalued facts.
- Dependency on the entire key.
- Independent vs. dependent facts.
- The presence of mutual constraints.
- The presence of nonunique or nonsingular representations.

Finally, the desirability of normalization has to be assessed in terms of its performance impact on retrieval applications.

Acknowledgments. I am very grateful to Ted Codd and Ron Fagin for reading earlier drafts and making valuable comments, and especially to Chris Date for helping to clarify some key points.

References

1. Codd, E.F. A relational model of data for large shared data banks. *Comm. ACM* 13, 6, (June 1970), 377–387.
2. Codd, E.F. Normalized data base structure: A brief tutorial. *ACM SIGFIDET Workshop on Data Description, Access, and Control*, Nov. 11–12, 1971, San Diego, California. E.F. Codd and A.L. Dean (Eds.).
3. Codd, E.F. Further normalization of the data base relational model. R. Rustin (Ed.), *Data Base Systems* (Courant Computer Science Symposia 6), Prentice-Hall, Englewood Cliffs, N.J., 1972. Also IBM Research Report RJ909.
4. Date, C.J. *An Introduction to Database Systems*. 3rd Ed. Addison-Wesley, Reading, MA, 1981.
5. Fagin, R. Multivalued dependencies and a new normal form for relational databases. *ACM Transactions on Database Systems* 2, 3, (Sept. 1977). Also IBM Research Report RJ1812.
6. Fagin, R. Normal forms and relational database operators. *ACM SIGMOD International Conference on Management of Data*, May 31–June 1, 1979, Boston, Mass. Also IBM Research Report RJ2471, Feb. 1979.
7. Kent, W. A primer of normal forms. *IBM Technical Report TR02.600*, Dec. 1973.
8. Ling, T.-W., Tompa, F.W., and Kameda, T. An improved third normal form for relational databases. *ACM Transactions on Database Systems* 6, 2, (June 1981), 329–346.

CR Categories and Subjects: H.2.1 [**Database Management**]: Logical Design—*normal forms*
General Term: Design

Received 3/82; revised 6/82; accepted 9/82

Computing Practices readers are directed to this month's Forum letter, "On Animals Around the 'Wolf Fence'" by Ted Tenny, which refers to the Procnique by E.J. Gauss in the November 1982 issue, and to the Technical Correspondence "On File Comparison Techniques" by Arthur I. Schwarz, referring to the Computing Practices article by Michael Pechura, which appeared in September 1982.

SELECTED BIBLIOGRAPHY

Appleton, D. S. Data Resource Techniques. Revision 2.0.
California: D. Appleton Company, Inc., 1987.

BeDell, Eugene F. The Computer Solution.
Illinois: Dow Jones-Irwin, 1985.

Durell, William R. Data Administration.
New York: McGraw-Hill, Inc., 1985

Holland, Robert H. Database: A Manager's Guide.
California: Technology Transfer Institute. 1988.

Lucas, Hendry C. Jr. Toward Creative System Design.
New York and London: Columbia University Press, 1974

LohMuller, Keith. Introduction to Business Programming and
System Analysis. Pennsylvania: Tab Books, Inc., 1983

Martin, James. Principles of Database Management.
New Jersey: Prentice-Hall, Inc., 1976.

Martin, James. Strategic Data-Planning Methodology.
New Jersey: Prentice-Hall, Inc., 1982.

Martin, James and Hershey, E. A. III.
Information Engineering Management White Paper.
Georgia: Knowledgeware, Inc., 1986

McClure, Carma. The CASE for Structured Development.
Technical PC Journal, vol. 6, No. 8, 1988.

McLeod, Raymond, Jr. Management Information Systems. 3rd ed.
Chicago: Science Research Associates, Inc., 1986

Rifin, Glenn. Northrop Designs Own Systems To Answer Pressure
To Integrate. Computer World, 12 September 1988.

Spaccapietra, Stefano. Entity-Relationship Approach: TEN Years of Experience in Information Modeling. 5th ed. New York: Elsevier Science Publishing Company, Inc. 1986.

U. S. Government Printing Office. Managing Computer Systems Development. Federal Registrar. AD-A158976, 1985